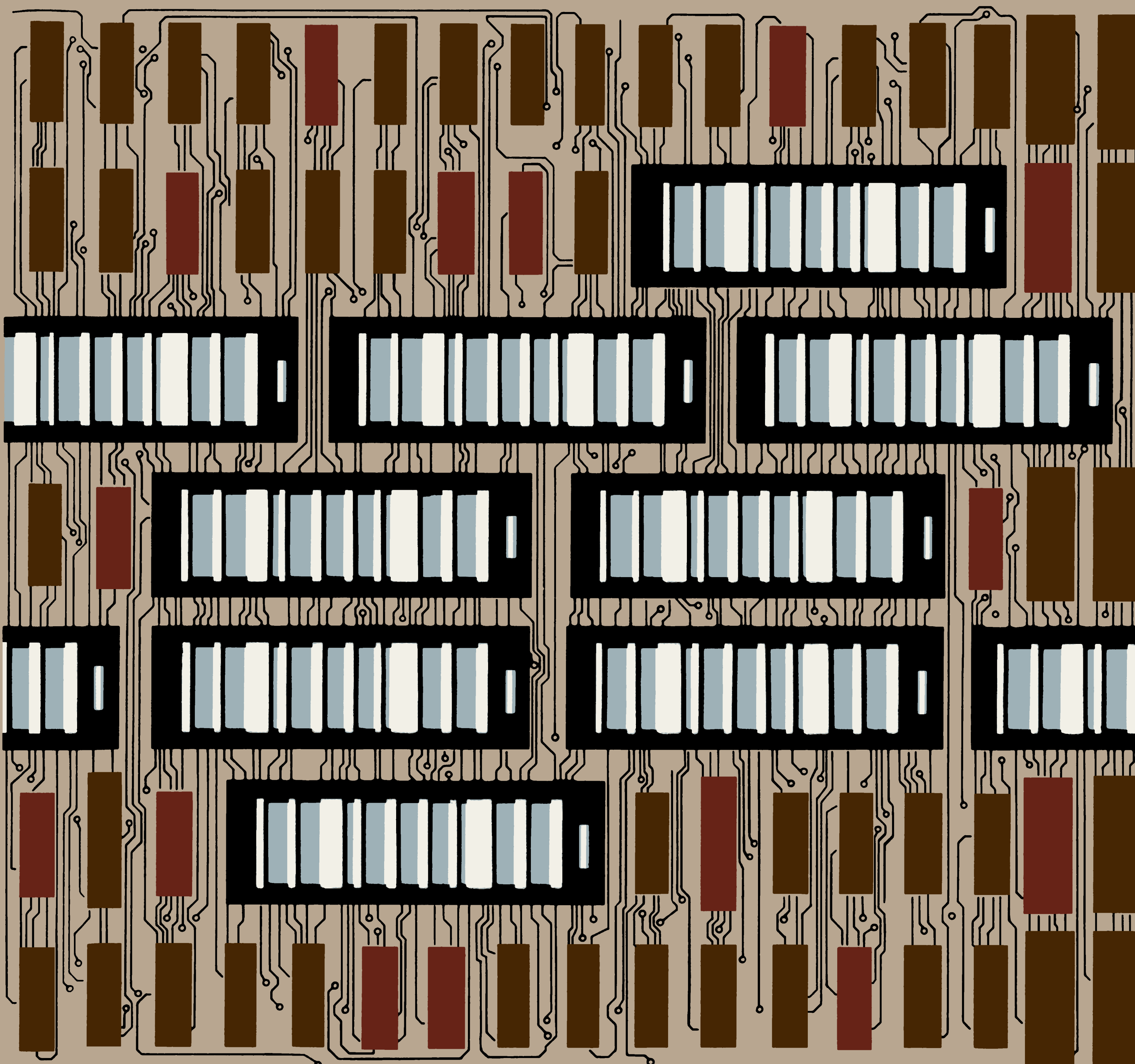
VAX-11/750

Diagnostic System Overview Manual



digital

BLANK

EK-VXD75-UG-002

VAX-11/750 Diagnostic System Overview Manual

digital equipment corporation • maynard, massachusetts

First Edition, August 1980
Second Edition, May 1981

Printed in U.S.A.

This document was set on DIGITAL's DECset-8000 computerized typesetting system.

CONTENTS

CHAPTER 1	**INTRODUCTION**	1-1
1.1	SCOPE	1-1
1.2	VAX-11/750 DIAGNOSTIC SYSTEM STRUCTURE	1-2
1.3	DIAGNOSTIC STRATEGY	1-5
1.4	VAX-11/750 REMOTE DIAGNOSIS MODULE	1-5
CHAPTER 2	**CONSOLE COMMANDS AND PROCEDURES**	2-1
2.1	FRONT PANEL CONTROLS AND INDICATORS	2-2
2.1.1	Five Position Keylock Switch	2-2
2.1.2	Power On Action Switch	2-2
2.1.3	Boot Device Switch	2-3
2.1.4	RESET	2-3
2.1.5	CPU State Indicator Lights	2-3
2.1.6	Remote Diagnosis Indicators	2-3
2.2	CONSOLE COMMAND LANGUAGE	2-3
2.2.1	Typing Errors and Illegal Characters	2-4
2.2.2	Control Characters	2-5
2.2.2.1	CTRL/P	2-5
2.2.2.2	CTRL/U	2-5
2.2.2.3	CTRL/D	2-5
2.2.3	Console Commands	2-5
2.2.3.1	Boot Command	2-5
2.2.3.2	Continue Commands	2-7
2.2.3.3	Deposit and Examine Commands	2-7
2.2.3.4	Index of Examines and Deposits	2-8
2.2.3.5	Initialize Command	2-10
2.2.3.6	Halt Command	2-10
2.2.3.7	Next Command	2-11
2.2.3.8	Start Command	2-11
2.2.3.9	Test Command	2-12
2.2.3.10	X Command, Binary Load/Unload	2-13
2.3	TU58 CARTRIDGE TAPE DRIVE	2-13
CHAPTER 3	**MICRO VERIFY PROGRAM**	3-1
3.1	INVOKING MICRO VERIFY	3-1
3.2	MICRO VERIFY ERROR MESSAGE INTERPRETATION	3-1
3.3	MICRO VERIFY TEST STRUCTURE	3-4
3.3.1	Bus and Scratch Pad Tests (@, C, E, F, I, J, L)	3-4
3.3.2	XB, IR, and OSR Test (O)	3-4
3.3.3	Source XB PC Increment Test (Q)	3-4
3.3.4	RNUM/DSIZE Test (R, T)	3-4
3.3.5	Parity Checker Test (X,[,])	3-5
3.3.6	Cache Test (Λ)	3-5
3.4	MICRO VERIFY ERROR MESSAGE FAILURES	3-5
CHAPTER 4	**BOOT AND RESTART FUNCTIONS**	4-1
4.1	BOOTING WITH THE FRONT PANEL SWITCHES	4-1
4.1.1	Booting VMS Automatically	4-1
4.2	BOOTING WITH CONSOLE COMMANDS	4-2
4.3	VAX-11/750 COLD START	4-3
4.3.1	Cold Start Sequence	4-3
4.3.1.1	Console Subsystem Action on Boot	4-3

CONTENTS (Cont)

4.3.1.2	Device ROM Code Function on Boot	4-5
4.3.1.3	Boot Block Code Operation	4-5
4.3.1.4	VMB Operation	4-6
4.3.1.5	SYSBOOT and VMS Operation	4-6
4.3.1.6	DIAGBOOT and Supervisor Operation	4-6
4.4	BOOT58 OPERATION	4-7
4.4.1	Booting the BOOT58 Command Processor	4-7
4.4.2	BOOT58 Commands	4-8
4.4.3	BOOT58 Command Parameters	4-8
4.5	BOOTING FROM THE TU58 WITHOUT BOOT58	4-9
4.6	WARM START	4-9
4.6.1	Console Subsystem Action on a Warm Start	4-10
4.6.2	VMS Action on a Warm Start	4-11
CHAPTER 5	**RUNNING LEVEL 4 DIAGNOSTIC PROGRAMS**	5-1
5.1	BOOTING LEVEL 4 DIAGNOSTIC PROGRAMS	5-1
5.2	LEVEL 4 PROGRAM ERROR INTERPRETATION AND LOOP CONTROL	5-2
CHAPTER 6	**PRIMARY AND SECONDARY LOAD PATHS**	6-1
6.1	PRIMARY LOAD PATH BOOT (STANDALONE)	6-1
6.2	SECONDARY LOAD PATH BOOT (STANDALONE)	6-1
6.3	DIAGNOSTIC SUPERVISOR ATTACH COMMAND SPECIFIC TO THE VAX-11/750	6-4
6.4	RUNNING THE SUPERVISOR ON-LINE	6-5
CHAPTER 7	**BUILDING AND MAINTAINING THE DIAGNOSTIC SYSTEM DISK**	7-1
7.1	BUILDING A DIAGNOSTIC DISK PACK	7-1
7.1.1	MASSBUS Single Disk Systems	7-1
7.1.2	MASSBUS Dual Disk Systems	7-1
7.2	UPDATING DIAGNOSTIC FILES	7-3
CHAPTER 8	**VAX-11/750 TROUBLESHOOTING GUIDELINES**	8-1

FIGURES

1-1	VAX-11 Diagnostic System User Documentation	1-1
1-2	VAX-11/750 Diagnostic System Load and Control Sequences	1-5
4-1	Boot Control Flow and Options	4-4
4-2	Restart Parameter Block, First Four Longwords	4-10

TABLES

1-1	Related Manuals	1-2
2-1	Console Command Syntax and Semantics	2-4
2-2	Console Command Error Codes	2-5
2-3	Software Boot Control Flags	2-6

TABLES (Cont)

2-4	Boot Device Codes	2-6
2-5	Console Halt Codes	2-11
3-1	Micro Verify Error Codes	3-2
4-1	Device ROM Code Starting Addresses	4-5
6-1	Device Naming Conventions	6-4

BLANK

CHAPTER 1
INTRODUCTION

1.1 SCOPE

This manual explains the VAX diagnostic system as it applies to VAX-11/750 computer systems. It covers console commands, the micro verify program, boot and restart procedures, macro level diagnostics, diagnostic system maintenance, and troubleshooting techniques. The manual does not cover details of the diagnostic supervisor or diagnostic programs common to all VAX systems. And it does not explain microdiagnostics. The manual will serve as a reference for field service engineers and as a resource for field service, manufacturing, and customer training programs. In order to make good use of this manual, you should be familiar with VAX architecture, VAX/VMS software, and the VAX-11/750 hardware. In the examples throughout this manual, operator input is underlined.

This manual is part of a four level documentation set, as shown in Figure 1-1.

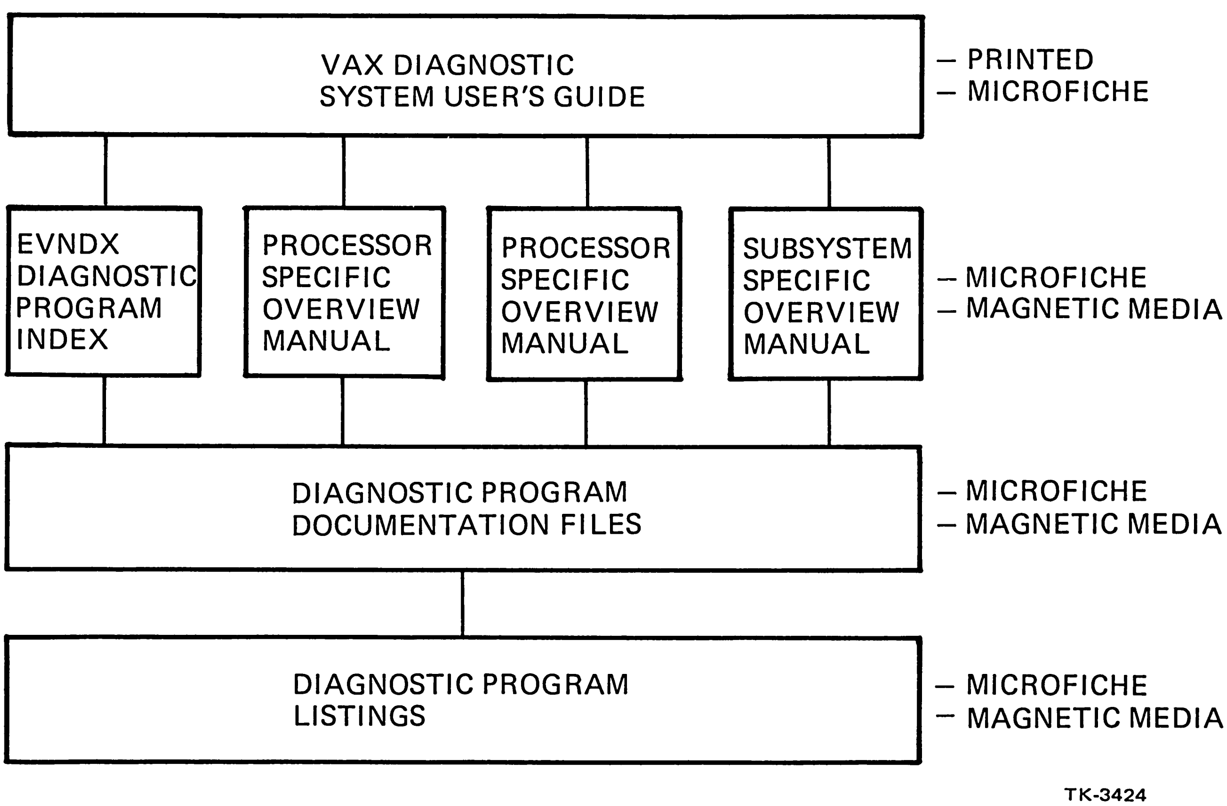


Figure 1-1 VAX-11 Diagnostic System User Documentation

These four levels form a progression in focus from general to specific. To apply the VAX diagnostic system effectively, you should become familiar with each documentation level. The *VAX Diagnostic System User's Guide* contains stable information that applies across all VAX computer systems. It explains the VAX diagnostic system structure and strategy and the various uses of the diagnostic supervisor. The *VAX Diagnostic System User's Guide* is available on hard copy and microfiche.

This manual, the *VAX-11/750 Diagnostic System Overview Manual*, is available on hard copy and on microfiche and ASCII files. The manual may be revised periodically. Revisions will be available on microfiche and magnetic media and distributed with other diagnostic system microfiche updates.

Program documentation files and program listings are available on microfiche and magnetic media. These are revised and distributed periodically. The program documentation files give information which help you effectively use the diagnostic programs.

DIGITAL also distributes EVNDX, the VAX Development MAINDEC Index, on microfiche and magnetic media. EVNDX enables users to keep up with the periodic changes and additions to the VAX-11 diagnostic system.

Table 1-1 lists related manuals.

Table 1-1 Related Manuals

Title	Order Number	Media
VAX Diagnostic System User's Guide	EK-VX11D-UG	Hard-copy
VAX-11/750 Installation and Acceptance Manual	EK-SI750-IN	Hard-copy and microfiche
VAX-11/750 Technical Descriptions:		
Central Processor Unit	EK-KA750-TD	Hard-copy and microfiche
UNIBUS Interface	EK-DW750-TD	Hard-copy and microfiche
Memory System	EK-MS750-TD	Hard-copy and microfiche
Power System	EK-PS750-TD	Hard-copy and microfiche
MASSBUS Adapter	EK-RH750-TD	Hard-copy and microfiche
Floating Point Accelerator	EK-FP750-TD	Hard-copy and microfiche
VAX-11 Architecture Handbook	EB-17580	Hard-copy
VAX-11 Software Handbook	EB-15485	Hard-copy

1.2 VAX-11/750 DIAGNOSTIC SYSTEM STRUCTURE

The VAX-11 diagnostic system consists of six program levels, as follows.

Level 1 – Operating system (VMS) based diagnostic programs

Level 2R – Diagnostic supervisor-based diagnostic programs restricted to running under VMS only

Certain peripheral diagnostic programs which are not supported by the supervisor in the standalone mode

System diagnostic control program

Level 2 – Diagnostic supervisor-based diagnostic programs that can be run either under VMS (on-line) or in the standalone mode

Bus interaction program

Formatter and reliability level peripheral diagnostic programs

Level 3 – Diagnostic supervisor-based diagnostic programs that can be run in standalone mode only

Functional level peripheral diagnostic programs
Repair level peripheral diagnostic programs
CPU cluster diagnostic programs

Level 4 – Standalone macrodiagnostic programs that run without the supervisor

Hard-core instruction test
Cache/ Translation Buffer test

Console Level – Console-based diagnostic programs that can be run in the standalone mode only.

The diagnostic programs which test peripheral devices are not processor specific. They run on VAX-11/750 processors and other VAX processors as well. Transportable programs use the letter V as the second character of the five-character program code (e.g., EVREA).

Of the diagnostic programs which test the VAX-11/750 CPU cluster, some are transportable and some are processor specific, as shown below. VAX-11/750 specific diagnostic programs use the letter C as the second character of the five-character program code (e.g., ECKAL).

- EVKAA Hard-core Instruction Test
  Level 4
  Transportable

EVKAA verifies the central processor functions that are essential to the operation of the diagnostic supervisor. It tests the instruction stream decoder, the data path logic, the condition code logic, address mode forks, and a set of instructions that are hard-core to the diagnostic supervisor. The program follows a building block structure so that a smaller amount of machine functionality is required at the beginning of the program than is required at the end.

Since the program executes macro level instructions, it tests the CPU microcode as well as the logic. Since it runs without the supervisor, it serves as a bridge between the micro verify program and the supervisor.

When the program detects an error, it executes a HALT instruction. See Chapter 5 for an explanation of how to proceed from an error halt.

- ECKAL Cache/Translation Buffer Diagnostic Program
  Level 4
  VAX-11/750 specific

The cache/translation buffer diagnostic program quickly verifies the cache and translation buffer portion of the VAX-11/750 central processor.

Like the hard-core instruction test, this program is written in MACRO-32 code. It also serves as a bridge between the micro verify program and the diagnostic supervisor. See Chapter 5 of this manual and the program documentation file for an explanation of how to use this program.

- ECKAM Memory Subsystem Test
  Level 3
  VAX-11/750 specific

  ECKAM detects and isolates VAX-11/750 memory subsystem failures. It tests the memory map, data bus, row select bus, address bus, error correction code logic, control/status register 0, the bootstrap ROMs, and the CPU XB error bit. In addition, it includes a CPU lost error test, a moving inversions test, and a moving inversions test with manual array selection.

  Cluster Exerciser (five programs):

- EVKAB Basic Nonprivileged Instruction Exerciser (Integer, Logical, and String Instructions)
  Level 2
  Transportable

- EVKAC Subsetable Instruction Exerciser (Floating-Point Instructions)
  Level 2
  Transportable

- EVKAD Compatibility Mode Instruction Exerciser
  Level 2
  Transportable

- EVKAE Privileged Architecture Exerciser
  Level 3
  Transportable

- ECKAX VAX-11/750 Specific Cluster Exerciser
  Level 3
  VAX-11/750 specific

  ECKAX tests all portions of the VAX-11/750 hardware that are not specified by the VAX-11 architecture.

- ECCBA Unibus Interface Diagnostic Program
  Level 3
  VAX-11/750 specific

  ECCBA tests the logic and functions of the VAX-11/750 UNIBUS Interface. IF PMK-05 is available, ECCBA will perform block transfer tests.

- ECCAA MASSBUS Adapter test
  Level 3
  VAX-11/750 specific

ECCAA tests the logic and functions of the MASSBUS adapter (RH750). If an RH11-TB MASSBUS exerciser is available, you can use Test 20 to test the RH750 on-line.

See the VAX Development MAINDEC Index (EVNDX) for a complete list of VAX diagnostic programs. If you have questions concerning the use of any specific diagnostic program, you will find a detailed explanation in the appropriate program documentation file.

The VAX-11/750 diagnostic system provides some flexibility concerning the load paths and execution control of different program levels. For example, the diagnostic supervisor and level 2 and level 3 programs can be loaded from the TU58 tape drive or from the system disk. If the primary mass storage load path does not work properly, you can use the TU58 to load in programs which will help you to repair the load path problem (see Chapter 6, Paragraph 6.2). Although level 2 programs are flexible and will run in the user mode or in the standalone mode, level 2R, 3 and 4 programs are less flexible. Figure 1-2 shows the load and control sequences and operating modes for the VAX-11/750 diagnostic system.

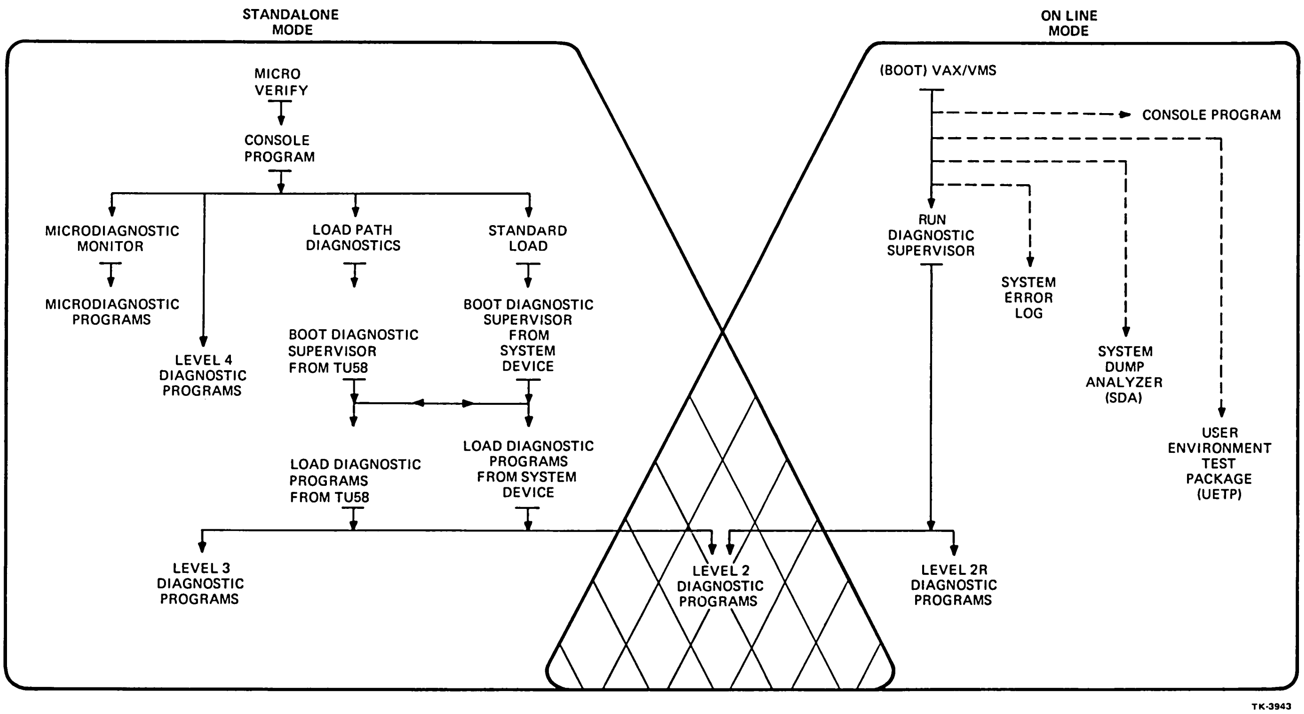


Figure 1-2 VAX-11/750 Diagnostic System Load and Control Sequences

1.3 DIAGNOSTIC STRATEGY

Remote diagnosis forms the basis of DIGITAL's strategy for VAX-11/750 system maintenance. For systems that have no remote diagnostic support, you can run diagnostics locally.

In general, you should run on-line diagnostics when possible, to identify the failing subsystem. Once you know what subsystem is at fault, run level 3 programs and then level 2 programs (bottom up approach) to isolate the failure to a field replaceable unit. If you cannot boot the diagnostic supervisor from the system disk, try to boot it from the TU58. If this fails, start with the hard-core instruction test. See Chapter 8 for troubleshooting guidelines.

1.4 VAX-11/750 REMOTE DIAGNOSIS MODULE

A customer who buys a maintenance contract from DIGITAL will normally have a Remote Diagnosis Module (RDM) installed in the VAX-11/750 CPU backplane. This module remains the property of DIGITAL, and should be used only by DIGITAL field service engineers. When customers are requesting service, they should leave the machine state undisturbed.

BLANK

CHAPTER 2
CONSOLE COMMANDS AND PROCEDURES

The VAX-11/750 console is an important diagnostic tool. It enables the operator

- to start and stop the CPU instruction set processor.
- to examine and deposit locations in main memory, I/O registers, processor registers, and internal registers.
- to control CPU execution.

The console hardware consists of a dedicated console terminal, the switches and lights on the front of the CPU cabinet, the integral TU58 tape drive, and the console-processor interface. When the remote diagnosis module (RDM) is installed, it forms a part of that interface. The console program is implemented in microcode in the control store RDM in the processor.

The VAX-11/750 console runs in two modes: program I/O mode and console I/O mode. These modes are mutually exclusive. One of the two will always be enabled while there is power to the machine. In the program I/O mode (also known as the system terminal mode), the console functions like the other terminals on the VAX-11/750 system. In this mode, the console passes characters between the terminal and the instruction set processor (ISP) running in the CPU.

In the console I/O mode (also called console mode), the console program interprets and acts on commands typed on the console terminal. The VAX-11/750 will enter the console I/O mode whenever the instruction set processor halts.

A variety of conditions cause the CPU to halt and enter the console I/O mode if the front panel keyswitch is in the LOCAL or REMOTE position.

- Cycle power or press the RESET button with POWER ON ACTION switch in the HALT position.
- Cycle power or press the RESET button with POWER ON ACTION switch in the RESTART/HALT position if the restart attempt fails.
- Cycle power or press the RESET button with the POWER ON ACTION switch in the BOOT position if the cold start attempt fails.
- Cycle power or press the RESET button with the POWER ON ACTION switch in the RESTART/BOOT position if both the warm start and the cold start attempts fail.
- Cycle power or press the RESET button with the POWER ON ACTION switch in any position if the micro verify program fails.
- Boot, continue, or start console command failure.

- Operator types CTRL/P (∧P) on the console terminal.
- The instruction set processor executes a HALT MACRO-32 instruction. (See Table 2-5 for the HALT codes and their meanings.)

When the processor enters the console I/O mode, it types on the console terminal the address contained in the program counter (PC), a two digit code which identifies the reason for the halt, and the console prompt symbol, >>>. The prompt symbol shows that the console program is in the console I/O mode.

2.1 FRONT PANEL CONTROLS AND INDICATORS

The front panel of the VAX-11/750 has four switches and seven indicator lights. The switches and their functions are outlined below:

2.1.1 Five Position Keylock Switch

OFF	No power to the CPU (except optional battery backup and the time-of-year clock) or to memory.
LOCAL	The CPU responds to console commands, and the remote diagnosis line is completely disabled.
LOCAL/SECURE	Console commands are ignored. The remote diagnosis line is completely disabled, and the RESET switch is disabled.
REMOTE	The console functions are enabled and may be activated from the remote line only. However, the remote line has the capability to return control to the local terminal.
REMOTE/SECURE	Console commands are ignored. The remote line replaces the console terminal, and the RESET switch is disabled. With the keyswitch in this position, the remote diagnosis module may not be used to identify the source of system failures. RDM commands are also ignored.

2.1.2 Power On Action Switch

This four-position rotary switch controls the machine on a power up sequence.

BOOT	Attempt to cold start the VAX-11/750 system (VMS or BOOT58). If the cold start fails (the cold start flag is set), halt the CPU. Prompt for a console command if the keyswitch is in LOCAL or REMOTE.
RESTART/BOOT	Attempt to warm start the VAX-11/750 system (VMS only). If the warm start fails (the warm start flag is disabled, or the memory does not contain a valid restart parameter block <RPB>), attempt to cold start the system. If the cold start fails (the cold start flag is set), halt the CPU. Prompt for a console command if the keyswitch is in LOCAL or REMOTE.
HALT	Do not attempt to start the system. Prompt for a console command if the keyswitch is in LOCAL or REMOTE.
RESTART/HALT	Attempt to warm start the VAX-11/750 system (VMS only). If the warm start fails (the warm flag is disabled, or the memory does not contain a valid restart parameter block <RPB>), halt the CPU. Prompt for a console command if the keyswitch is in LOCAL or REMOTE.

2.1.3 Boot Device Switch

This four-position rotary switch selects the device to boot from. The VAX-11/750 memory controller contains four sockets for built-in Read Only Memory chips (ROMs) that contain the MACRO-32 code needed to bootstrap a device. The switch selects which of the four ROMs is to provide the bootstrap code when a boot sequence is initiated. Typically, this switch is left in the position corresponding to the VAX/VMS system disk. Position D always causes a boot from the console TU58 tape cartridge.

2.1.4 RESET

This pushbutton switch simulates a system power down sequence followed by a power up sequence. The system will come up in the state selected by the POWER ON ACTION switch. It is usually used only if the machine appears to be hung and does not respond to console commands. RESET does not actually cause a power loss, so memory remains unmodified.

2.1.5 CPU State Indicator Lights

POWER — This green light, when lit, indicates that DC power is present in the CPU and that the keyswitch is not in the OFF position.

RUN — When lit, this green light indicates that the machine is in the RUN state.

ERROR — When lit, this red light indicates that the machine is stopped because of a fatal control store parity error. To reset the machine and reenable the console commands, press the RESET button.

2.1.6 Remote Diagnosis Indicators

The following four lights are back-lit words, illuminated during various stages of remote diagnosis procedures. It should be noted that the REMOTE D, RD CARRIER, RD TEST, and RD FAULT lights only apply to systems having the optional remote diagnosis module installed by field service.

REMOTE D — Remote diagnosis (RD) software illuminates this green light whenever the keylock switch is in one of the two remote positions.

RD CARRIER — This amber light is lit by the RD software whenever it detects the presence of the remote port carrier. It indicates that the DIGITAL Diagnosis Center (DDC) has established connection.

RD TEST — The remote diagnosis software illuminates this green light while tests are in progress.

RD FAULT — This red light is lit by the Remote Diagnosis Module (RDM) if the RDM detects a fault in its own logic. No tests (except the RDM tests) should be attempted when the fault indicator is lit. However, you can attempt to run the VAX-11/750 system if you observe proper precautions.

2.1.7 Console Lamp Test

There is a lamp test pushbutton under the operator control panel to test all the lamps on the control panel. Open the front door to access lamp test button.

2.2 CONSOLE COMMAND LANGUAGE

The console command language enables the user to communicate with the VAX-11/750 hardware and microcode from the console terminal. A single letter with optional modifiers specifies a command. When the CPU is not executing instructions, it is halted, and the console terminal is said to be in console I/O mode. In this mode, the CPU is receptive to console commands and can perform the functions listed above. Direct memory access (DMA) activity to memory can also occur with the terminal in console mode, and all DMA transfers in progress continue even when the CPU is halted.

However, interrupts will not be serviced while the machine is halted. They will be serviced following a continue command from the console. Table 2-1 shows the symbols used in the console command descriptions that follow.

Table 2-1 Console Command Syntax and Semantics

Symbol	Function
< >	Angle brackets denote category names. For example, the category name <address> may be used to represent any valid address, instead of actually listing all the strings of characters that can represent an address.
[ ]	Parts of expressions in brackets do not need to be typed if defaults are to be used.
<space>	Represents one typed space.
<address>	Represents an address argument. Valid types are explained in the following pages in relation to specific commands. However, virtual addresses that reference nonexistent or nonresident pages will cause the console to abort execution of the console command which referenced that address. An appropriate error message will be displayed.
<data>	Represents a numeric argument.
<qualifier>	Represents a command modifier (also called a switch). Valid <qualifier> types are explained later in relation to specific commands.
<input-prompt>	Represents the console's input prompt string >>>.
<CR>	Carriage return, line feed.

2.2.1 Typing Errors and Illegal Characters

You may correct typing errors (before a <CR> causes the CPU to execute the command) using the DELETE key. When you press the DELETE key, the console will echo the character being deleted (after printing a backslash (\) upon receipt of the first deletion). The console will also add a backslash between the last deletion and the next input character.

For example:

```
operator types      12er<del><del>34

console prints      12er\re\34

console sees        1234
```

Example 2-1 Character Deletion

The console attempts to interpret each character as it is typed. If the console cannot interpret the next character in the context of the current command, it sounds a "beep" (bell) and ignores the character. This does not abort the entire command. Instead, the command may be completed by typing the correct character(s). If you type a command which is syntactically correct but semantically invalid, the console responds with an error code. Table 2-2 lists the console command error codes.

Table 2-2 Console Command Error Codes

Code	Meaning
11	Illegal access to an IPR.
20	Access Violation, Translation Not Valid, or Machine Check during a read or write.
30	Binary transfer checksum error.
33	Unrecognizable boot device.
34	Controller not A,B,C, or D in boot command.

2.2.2 Control Characters

2.2.2.1 CTRL/P – This is typed by holding the CONTROL key while typing a P. This command puts the machine in console I/O mode, halts the processor (see the halt command), and causes the console terminal to type a halt message (<PC><space>02). Typing a CTRL/P while in console I/O mode causes the console defaults to be reinitialized and the halt message to be typed on the console terminal.

```
                                      ! Processor is running.
^P                                    ! Enter console I/O mode.
>>>                                   ! Console prompt.
```

Example 2-2 CTRL/P Command

2.2.2.2 Control U – CTRL/U tells the system to ignore all characters typed since the last carriage return. The console responds by typing

\<CR><input-prompt>.

2.2.2.3 CTRL/D – CTRL/D causes the console to enter the RDM command mode from the console I/O mode, if the remote diagnosis module is installed.

2.2.3 Console Commands

2.2.3.1 Boot Command

B[/X][/<n>][<space><ddcu>]<CR>

This command boots the operating system or diagnostic software from the device specified. The hexadecimal number, <n>, specifies the boot control flags. The command deposits this number in Register R5 before booting. For example, 200 sets bit 9 in R5, specifying boot control flag 9. <n> may be

from one to four digits in length. If <n> is omitted, the default value for R5 is 0. Table 2-3 shows the software boot control flags.

If you type/X, the boot command inhibits micro verify. The central processor is initialized, however. This means that the micro verify tests are normally run at the beginning of the boot sequence.

<ddcu> represents the boot device, where <dd> is a two letter device mnemonic. Table 2-4 shows the boot device codes. <c> specifies the I/O channel adapter. A,B,C, and D are possible values. <u> is a one digit number that specifies the device drive number. If you do not specify <ddcu>, the console checks the BOOT DEVICE switch on the front panel to determine the fault boot device. If you specify boot control flags with <n>, you must also specify <ddcu>. Otherwise, the boot control flags are ignored.

Table 2-3 Software Boot Control Flags

Flag	Hex Value	Function
0	1	Conversational boot. At various points in the system boot procedure, parameters and other inputs are solicited from the console.
1	2	Debug. This flag is passed through to VMS and causes the code for the executive debugger to be included in the running system.
2	4	Initial breakpoint. If this flag is set, and the executive debugger code is included (flag bit 1), then a breakpoint will occur immediately after the exec enables mapping.
3	8	Not used on the VAX-11/750.
4	10	Diagnostic boot. This flag causes a boot by file name for the diagnostic supervisor.
5	20	Bootstrap breakpoint. This flag causes the bootstrap to stop at a breakpoint after performing necessary initialization.
6	40	Image header. If this flag is set, the transfer address from the image header of the boot file is used. Otherwise control transfers to the first byte of the boot file.
7	80	Memory test inhibit. This flag inhibits the testing of memory during bootstrapping.
8	100	File name. Causes the bootstrap to solicit the name of the boot file.
9	200	Halt before transfer. Causes a halt instruction to be executed prior to the transfer to the secondary boot file. This option is useful for debugging purposes.

Table 2-4 Boot Device Codes

Device Code (dd)	Device Type
DL	RL02
DM	RK06/7
DB	RP04/5/6, RM03
DD	TU58

Following a successful boot, the console enters program I/O mode (system terminal mode). If micro verify has been executed (/X was not typed), the console also prints a message telling whether the test revealed any errors (see the test command).

2.2.3.2 Continue Command

C<CR>

The continue command restarts execution of a halted program at the address currently contained in the program counter. The CPU is not initialized, and the console terminal enters system terminal mode once the continue command is issued.

2.2.3.3 Deposit and Examine Commands

D[<qualifier-list>]<space><address><space><data><CR>
E[<qualifier-list>]<space><address><CR>

Deposit and examine commands are treated together because their formats are quite similar. Both commands require definition of the address space and the size of the operand in addition to the address. To make multiple examines or deposits easier, there is a system of defaults for each of the items that must be specified for these commands. Some of the defaults are automatic (such as longword size for general registers), and some are set up by the immediately preceding examine or deposit. All other console commands (except <CR>) cause the defaults to be set to their initial values.

Size qualifiers:

/B	byte
/W	word
/L	longword

Space qualifiers:

/V	virtual address
/P	physical address
/I	internal processor register (IPR)
/G	general processor register (GPR)

Address values:

nnnnnnnn	hexadecimal number
*	last location referenced
P	PSL
+	next location (deposit ONLY)

Data:

nnnnnnnn	hexadecimal number

These commands deposit (write) or examine (read) <data> at the <address> specified. The address space and size used depends upon the qualifier or qualifiers specified with the command. If no address space qualifier is used, the default is physical address space. Following another deposit or examine, the address space qualifier of the previous command is used as the default.

If no size qualifier is typed, the default for a physical or virtual deposit or examine is longword. However, following another deposit or examine, the size used in the previous command is the default. The size for an IPR or GPR deposit or examine is always longword, and these commands do not change the current size default.

<Address> must be from one to eight hexadecimal digits long. The initial default is zero. However, the default is unpredictable when the address space is changed. Following another virtual or physical deposit or examine, the default is the sum of the address from the last deposit or examine plus size (number or bytes) from the last deposit or examine. Typing a + for <address> (for deposit only) also gets this default. Following another IPR or GPR deposit or examine, the default is the address from the last deposit or examine plus one. Using a P for <address> causes a longword reference to the processor status longword, independent of the defaults for address space and size. Using P has no effect on any of the defaults.

<Data> must be represented by one to eight hexadecimal digits. If you supply more digits than the size specifies, the extra digits on the left are ignored. If you supply fewer digits, zeros are appended to the left.

Deposit response: a successful deposit always produces a console prompt.

Sample examine responses: (console input underlined)

```
>>>E/P 1234                          ! Examine physical
                                     ! address 1234.
P ØØØØ1234 ABCDEF89

>>>E/V 1234                          ! Examine virtual
                                     ! address 1234.

P ØØØØ5634 Ø1234567                  ! Note that virtual
                                     ! examines display the
                                     ! translated physical
                                     ! address.

>>>E/G Ø                             ! Examine general
                                     ! register RØ.
G ØØØØØØØØ 98765432

>>>                                  ! Console prompt.
```

Example 2-3 *Examine* Command

2.2.3.4 Index of Examines and Deposits

E<space>*<CR> ! Examine the location
! last referenced.

E<CR> ! Examine the next location (last
! address plus the size of default
! data type in bytes).

```
E<space><address><CR>              ! Examine <address>; all
                                   ! switches are defaulted to
                                   ! last examine or deposit.

E/G<space><address><CR>            ! Examine GPR; register number
                                   ! is <address>.<Address>must
                                   ! be a value from 0 to F
                                   ! hexadecimal.

E/I<space><address><CR>            ! Examine IPR; register number
                                   ! is <address>.

E/P<space><address><CR>            ! Examine physical <address>.

E/V<space><address><CR>            ! Examine virtual <address>.

E<space>P<CR>                      ! Examine PSL.

E/W/P<space><address><CR>          ! Examine a word at physical
                                   ! <address>.

E/P/W<space><address><CR>          ! Examine a word at physical
                                   ! <address>. Notice that /W
                                   ! and /P may be reversed.

E/L/V<space><address><CR>          ! Examine a longword at virtual
                                   ! <address>. /L and /V
                                   ! may be reversed.

D<space>*<space><data><CR >        ! Deposit <data> in the
                                   ! location last referenced

D<space>+<space><data><CR>         ! Deposit <data> in the next
                                   ! location (last location plus
                                   ! the size of the default
                                   ! type in bytes).

D/G<space><address>-               ! Deposit <data> in GPR
<space><data><CR>                  ! <address>.

D/I<space><address>-               ! Deposit data in IPR
<space><data><CR>                  ! <address>.

D/P<space><address>-               ! Deposit <data> in physical
<space><data><CR>                  ! <address>.

D/V<space><address>-               ! Deposit <data> in virtual
<space><data><CR>                  ! <address>.

D<space>P<space><data><CR>         ! Deposit <data> in PSL.

D<space><address>-                 ! Deposit <data> in
<space><data><CR>                  ! <address>. Switches are
```

	! defaulted to previous ! switches.
D/V/W<space><address>- <space><data><CR>	! Deposit a word of <data>in ! virtual <address>. /V and /W ! may be reversed.
D/L/P<space><address>- <space><data><CR>	! Deposit a longword of <data> ! in physical <address>. /L and ! /P may be reversed.

2.2.3.5 Initialize Command

\>>>I<CR>
\>>>

This command performs the following functions:

- Initialize the processor.
- Clear the translation buffer.
- Clear the cache.
- Set the program counter to 0.
- Set the virtual address register to 0.
- Set the processor status longword to a known state. (041F0000)
- Enable the cache.
- Disable memory management.
- Perform I/O initialization (UNIBUS INIT)

```
>>>I                                    ! Initialize the CPU.

>>>                                     ! Console prompt.
```

Example 2-4 Initialize Command

2.2.3.6 Halt Command

\>>>H<CR>\
\>>>

The halt command is implemented in the VAX-11/750 for the sake of consistency. It does not actually halt the CPU, since the CPU must be halted, in the console I/O mode, in order to respond to the halt console command. However, the halt command initializes the console defaults, as if a kernel mode halt had just been executed. The console also prints a halt message whenever the CPU executes an ISP level HALT instruction or if the CPU stops instruction execution for any reason.

Table 2-5 defines the various console halt codes.

Table 2-5 Console Halt Codes

Code	Meaning
1	Executed micro verify sequence (test command).
2	CTRL/P, halt, or single macroinstruction mode.
4	Interrupt stack not valid.
5	Double bus write error halt.
6	Halt ISP level instruction executed.
7	Vector <1:0> = 3. Halt at Vector.
8	Vector <1:0> = 2. WCS is disabled or not present.
A	Change mode instruction executed while on the interrupt stack.
B	Change mode instruction executed and vector <1:0> is not equal to 0.
11	Power up and connot find RPB. POWER ON ACTION switch is at RESTART/HALT.
12	Power up and warm start/flag is disabled. POWER ON ACTION switch is at RESTART/HALT.
13	Power up and connot find good 64K of memory.
14	Power up and booting, but bad boot ROM or no ROM.
15	Power up, and the cold start flag is disabled.
16	Power up halt. The POWER ON ACTION switch is in the HALT position.
FF	Micro Verify test failure.

2.2.3.7 Next Command

```
N<CR>
xxxxxxxx 02
>>>
```

The next command causes the CPU to execute one ISP level instruction. The CPU then halts and reenters the console I/O mode.

2.2.3.8 Start Command

S[<space><address>]<CR>

The start command is normally used to start execution of programs that run without the operating system and without the diagnostic supervisor. Start performs the equivalent of the following console commands:

1. Initialize the CPU.

2. Deposit <address> into the Program Counter (PC). If no address is specified, the current value of the PC is used.

3. Perform the continue function to begin ISP level program execution.

Programs which may be started this way must be loaded into main memory before the start command is given.

NOTE
If AC LO is asserted after entering console mode, system will hang after executing start or continue commands.

```
>>>S 1ØØØ                    ! Start the program that
                             ! begins at address 1ØØØ.
```

Example 2-5 Start Command

2.2.3.9 Test Command

T<CR>

This command runs the micro verify program and initializes the processor. The micro verify program types a percent sign (%) on the terminal when it starts the test.

If all the tests run successfully, micro verify types a second %. If micro verify detects a failure, it types a single error character and then halts the processor in the console I/O mode. When the processor halts, the console types an error code, in place of the PC, and a halt code (FF). The halt code shows that this halt is due to an error condition in micro verify. See Chapter 3 for details on micro verify.

Example 2-6 shows the console output for a successful micro verify run. Example 2-7 shows the console output for a failure.

```
>>>T                         ! Run Micro Verify and
                             ! initialize.

%%                           ! Successful test completion.
xxxxxxxx Ø1
>>>                          ! Console prompt.
```

Example 2-6 Successful Micro Verify Run

```
>>>T                                    ! Run Micro Verify and
                                        ! initialize.

%E                                      ! Scratch pad bit test
                                        ! failure. See Chapter 3 for
                                        ! error message
                                        ! interpretation

00000054   FF                           ! Micro Verify test failure.
                                        ! Console halt following Micro
                                        ! Verify.

>>>                                     ! Console prompt.
```

Example 2-7 Micro Verify Failure

2.2.3.10 X Command, Binary Load/Unload

DIGITAL manufacturing uses this command to transfer test files. The X command is not useful to customers or field service engineers.

NOTE

The console ignores illegal characters and echoes them as bell codes.

2.3 TU58 CARTRIDGE TAPE DRIVE

The TU58 cartridge tape drive is an important part of the console subsystem. Because the TU58 is connected directly to the CPU, it maintains the capability to load diagnostic programs even if some system components are inoperative. This feature significantly increases VAX-11/750 system reliability. The TU58 may also be used to boot the system, to load files into physical memory, and to store files which describe and execute site-specific bootstrap procedures.

The tape cartridge is preformatted to store 2048 records, each containing 128 bytes. The controller provides random access to any record. The TU58 searches at 60 inches per second (ips) to find the file requested, then reads at 30 ips. Data read from the tape are verified through checksums at the end of each record or header. All data transfers between the TU58 and the CPU are implemented in 512-byte blocks, with the TU58 concatenating four 128-byte records to make one block. Data are transferred to the CPU at approximately 2 kilobytes per second.

BLANK

CHAPTER 3
MICRO VERIFY PROGRAM

The Micro Verify tests checks the VAX-11/750 central processor. Although micro verify does not test the CPU thoroughly, it doės test a large portion of the boot and diagnostic load path. The program should aid the user in identifying failures which prevent loading and execution of the hard-core instruction test (EVKAA) and the diagnostic supervisor.

3.1 INVOKING MICRO VERIFY

There are several ways to run micro verify. First, micro verify runs automatically on power up. Second, it runs automatically when the operator types the boot command (B) on the console, unless the operator qualifies the boot command with the /X qualifier. Third, the test command (T) on the console invokes micro verify.

If micro verify detects an error, it prints a message on the console terminal (see Paragraph 3.2), displays the console input prompt, and returns control to the console I/O mode in the console program.

3.2 MICRO VERIFY ERROR MESSAGE INTERPRETATION

Like the console software, the micro verify program is implemented entirely in the control store ROM in the central processor. You can find the code for the program in the VAX-11/750 microcode listing.

However, it is not necessary to read the program listing in order to make good use of micro verify. When the program starts, it prints a percent sign, %, on the console terminal. If this is followed by a second percent sign, <CR>, and a console prompt, no error has been detected. See Example 3-1.

```
>>>T                                ! Run Micro Verify and
                                    ! initialize.

%%                                  ! Successful test completion.
xxxxxxxx Ø1
>>>                                 ! Console prompt.
```

Example 3-1 Successful Micro Verify Run

If micro verify does detect an error, it attempts to print a one character error code following the starting percent sign, <CR>, a three digit code in place of the PC, a halt code, another <CR>, and a console prompt. See Example 3-2.

```
>>>T                                     ! Run Micro Verify and
                                         ! initialize.

%E                                       ! Scratch pad bit test
                                         ! failure. See Table 3-1 for
                                         ! error code interpretation.

00000052 FF                              ! Error clearing GPR.
                                         ! Console halt following
                                         ! Micro Verify.
>>>                                      ! Console prompt.
```

Example 3-2 Micro Verify Failure

If no character follows the first percent sign, it shows that there is a serious processor problem, a terminal problem, or a baud rate problem.

Failures in the data path often show up as failures in some other test. Accurate isolation of data path failures is impossible, because of limitations on program size. Therefore, the tests are written with the assumption that most of the data path is working. Error reports are based on this assumption.

Many failure modes prevent any error reporting by micro verify. When the program detects an error, it attempts to perform the following sequence.

1. Load the console halt error code.
2. Load the test error number into the PC.
3. Send a single error character to the console terminal.
4. If step 3 fails, loop, driving the PC contents onto the W bus.
5. If step 3 succeeds, halt.
6. The halt sequence then prints the PC, the error halt code, and the console prompt on the terminal.

The single character error codes have been chosen so that a one bit error in the character produces an invalid error code. For example, @ (60 hexadecimal) is legal; A (61 hexadecimal) is not legal.

Table 3-1 lists the error codes for all micro verify tests.

Table 3-1 Micro Verify Error Codes

CODE	PC+2	TEST NAME/ERROR MESSAGE
@		RBUS, WBUS Test
	000	bad bit in DREG or SUPROT
	001	bad bit in RBUS or WBUS
C		MBUS Test
	031	bad bit in QREG
	032	bad bit in MBUS

Table 3-1 Micro Verify Error Codes (Cont)

CODE	PC+2	TEST NAME/ERROR MESSAGE
E		Scratch Pad Bit Test
	051	error clearing RTEMP
	052	error filling RTEMP with ones
	054	error clearing GPR
	057	error filling GPR with ones
	058	error clearing IPR
	05B	error filling IPR with ones
	05D	error clearing MTEMP
	05E	error filling MTEMP with ones
F		MTEMP Explicit Address Test
	061	error addressing MTEMP0
	062	error addressing MTEMP1
	064	error addressing MTEMP2
	067	error addressing MTEMP4
	068	error addressing MTEMP8
I		RTEMP Explicit Address Test
	091	error addressing RTEMP0
	092	error addressing RTEMP1
	094	error addressing RTEMP2
	097	error addressing RTEMP4
	098	error addressing RTEMP8
J		IPR Explicit Address Test
	0A1	error addressing IPR0
	0A2	error addressing IPR1
	0A4	error addressing IPR2
	0A7	error addressing IPR4
	0A8	error addressing IPR8
L		GPR Explicit Address Test
	0C1	error addressing R0
	0C2	error addressing R1
	0C4	error addressing R2
	0C7	error addressing R4
	0C8	error addressing R8
	0CE	error addressing dual port
O		XB/IR/OSR Bit Test
	0F1	error in XB<31:0>
	0F2	error in XB<63:32>
	0F4	error in IR
	0F7	error in OSR
Q		Source XB PC Increment Test
	111	error in sourcing one byte from XB
	112	error sourcing 2 bytes from XB or incrementing PC by 1
	114	error sourcing an unaligned longword or incrementing PC by 2
	117	error incrementing PC by 4
R		RNUM/DSIZE Test
	121	error reading DSIZE ROM operand 1
	122	error loading/reading RNUM
	124	error reading DSIZE ROM operand 2
	127	error loading/reading RNUM
	128	error reading DSIZE ROM operand 3
	12B	error loading/reading RNUM
	12D	error reading DSIZE ROM operand 4
	12E	error loading/reading RNUM

Table 3-1 Micro Verify Error Codes (Cont)

CODE	PC+2	TEST NAME/ERROR MESSAGE
T		RNUM/DSIZE Test Continued
	141	error reading DSIZE ROM operand 5
	142	error loading/reading RNUM
	144	error reading DSIZE ROM operand 6
X		Cache Parity Error Test
	181	failed to get cache parity error
	182	bad Machine Check Error Summary Register
	184	bad Cache Error Register
[		TB Parity Error Test
	1B1	failed to get group 0 TB parity error
	1B2	bad TB Group Parity Error Register
	1B4	bad Machine Check Error Summary Register
	1B7	failed to get group 1 TB parity error
	1B8	bad TB Group Parity Error Register
	1BB	bad Machine Check Error Summary Register
]		Control Store Parity Error Test
	1D1	failed to get control store parity error
	1D2	error in control store parity error
^		Cache Test
	1E1	error filling cache with ones; location not initially = 0
	1E2	error filling cache with ones; unable to write ones

3.3 MICRO VERIFY TEST STRUCTURE

3.3.1 Bus and Scratch Pad Tests (@,C,E,F,I,J,L)

This section checks for bits stuck at one and stuck at zero on all internal 32-bit buses and registers, as follows:

- Float a zero in a field of ones.
- Float a one in a field of zeros.

In addition, the tests check the scratch pad register addressing by writing each register with its address and then reading it back.

3.3.2 XB,IR, and OSR Test (O)

This section verifies that a byte, a word, and a longword can be sourced from the Execution Buffer (XB) and that the PC is incremented properly. The XB (all 64 bits), the Instruction Register (IR), and the Operand Specifier Register (OSR), are all tested with the floating ones pattern.

3.3.3 Source XB PC Increment Test (Q)

This section tests the auto-incrementing of the PC when fetching ISTREAM data. The tests fetch a byte, a word, a longword, and then another byte from the XB. The tests then check the information and the state of the PC.

3.3.4 RNUM/DSIZE Test (R,T)

This section verifies the operation of the DSIZE ROM and the DSIZE latch. It also tests whether RNUM is loaded with the register number of the operand specifier when the operand specifier is loaded.

3.3.5 Parity Checker Test (X,[,])
This section tests the parity checking logic for the cache, translation buffer, and control store. The tests use special microcode functions to write bad parity in the cache, and in both sides of the translation buffer. When these locations (and a special control store location with bad parity) are referenced, micro traps should occur. The test sets a micro code flag to regain control from the machine check micro traps.

3.3.6 Cache Test (∧)
This section performs a cache addressing test. It also tests every location in the cache, one bit at a time, with floating one and floating zero patterns.

3.4 MICRO VERIFY ERROR MESSAGE FAILURES
In some cases, the micro verify program may print one percent sign and then <CR><input-prompt> on the console terminal. This probably indicates that the CPU has executed a micro trap after the program started. The micro trap has then returned control to the console program.

At this point, some console commands, such as memory examine or deposit, may not work properly, although the same commands worked perfectly before running micro verify. Command failures of this type occur when micro verify terminates improperly, leaving the CPU in an undefined, partially disabled state. Type I on the console terminal to reinitialize the CPU and restore console functionality.

BLANK

CHAPTER 4
BOOT AND RESTART FUNCTIONS

The VAX-11/750 system will react to a software crash, or a power up sequence according to the setting of the front panel switches. The console microcode reads the switches and determines whether to perform a cold start (boot), perform a warm start (restart), or halt the system. If the keyswitch is in LOCAL or REMOTE when the machine halts, the console subsystem enters the console I/O mode. Then the local or remote operator can use the console command language to boot the system.

4.1 BOOTING WITH THE FRONT PANEL SWITCHES

The console microcode examines the status of the front panel POWER ON ACTION switch and the BOOT DEVICE switch in four cases: when you apply power initially by turning the keyswitch, after recovery from a power failure, after a software crash, or if the operator presses the RESET button. The switches should be set to reflect the requirements of the user and the configuration of the system. The console subsystem interprets the four positions of the POWER ON ACTION switch and the console warm and cold start flags as explained in Chapter 2, Section 2.1.2.

At most, the console subsystem attempts one warm start and one cold start after a system crash or a power failure. Before attempting a warm start or a cold start, the console subsystem disables the warm or cold start flag as appropriate, to indicate that a warm or cold start is in progress. If the attempt to warm or cold start fails, the console subsystem again examines the flags and the POWER ON ACTION switch to determine whether to attempt a cold start or to halt the machine. If the machine halts and the keyswitch is in LOCAL or REMOTE, the console prompts for input. If the key switch is in a secure position, the machine halts without prompting for input. The front panel RUN light is extinguished.

When VMS boots successfully, the software issues a command, requesting the console subsystem to enable the cold start flag. In the same way, when VMS restarts successfully, the software issues a command to request the console subsystem to enable the warm start flag. If the cold start attempt fails, press the RESET button to initialize the machine before attempting another cold start.

4.1.1 Booting VMS Automatically

To ensure that VMS comes up when you boot the VAX-11/750 system automatically on power up, you must meet the following conditions.

1. Disk drive 0 on MBA0 or UBI0 is loaded with a disk pack containing a boot block, VMB.EXE, SYSBOOT.EXE, and VMS.

2. A boot device ROM corresponding to the VMS disk is installed in slot A, B, C, or D.

3. The BOOT DEVICE switch points to the appropriate boot ROM.

4. The POWER ON ACTION switch points to BOOT.

4.2 BOOTING WITH CONSOLE COMMANDS

The VAX-11/750 console subsystem runs in the console I/O mode when the CPU is halted and the keyswitch is in the LOCAL or REMOTE position. The console terminal displays a console prompt symbol, >>>.

Before you type the boot command, you must meet the following conditions to boot VMS or the diagnostic supervisor.

1. a. A disk drive attached to the VAX-11/750 system through any channel adapter is loaded with a diagnostic disk or a VMS disk.

   A diagnostic disk contains a boot block and the SYSMAINT directory. The SYSMAINT directory consists of

VMB.EXE	the primary bootstrap
DIAGBOOT.EXE	the secondary bootstrap
ECSAA.EXE	the diagnostic supervisor
Diagnostic program files	

   A VMS disk pack contains a boot block and the SYSEXE directory. The SYSEXE directory consists of

VMB.EXE	the primary bootstrap
SYSBOOT.EXE	the secondary bootstrap
VMS	the operating system files

   A TU58 tape cartridge contains a boot block and the file to be booted.

   b. Or, an appropriate TU58 tape cartridge is inserted in the TU58 tape drive on the front panel.

2. A boot device ROM corresponding to the disk is installed in slot A, B, C, or D.

3. The BOOT DEVICE switch points to the appropriate ROM.

The boot command takes the following format (see Section 2.2.3.1 of Chapter 2 for details).

B[/X][/n][<space><ddcu>[<CR>

Type /X if you do not wish to execute the micro verify program before booting. Omit the /n qualifier if you do not wish to use the options provided by the software boot control flags. Type /10 for /n to boot the diagnostic supervisor. See Table 2-3 in Chapter 2 for details on how to modify the boot process with the software boot control flags. For <dd> in <ddcu>, type the two character device code of the boot device (see Table 2-4 in Chapter 2).

The device code you type in identifies the device ROM code which brings in the boot block. The console subsystem matches the device code you type in against the first two bytes (reversed) of the device ROM code. For <c> in <ddcu> type A, B, C, or D, to identify the channel adapter to which the boot device is attached. And for <u> in <ddcu> type the drive number of the boot device.

Example 4-1 shows two typical boot commands.

```
>>>B/X DMAØ                      ! Boot VMS from
                                 ! the RKØ6 on channel adapter A
                                 ! drive Ø without running micro
                                 ! verify.
        .
        .
        .
>>>B/1Ø DBB1                     ! Boot the diagnostic
                                 ! supervisor in the
                                 ! standalone mode
                                 ! from the RPØ6
                                 ! on channel adapter
                                 ! B drive 1 after running
                                 ! micro verify.
```

Example 4-1 Boot Commands

4.3 VAX-11/750 COLD START

The console subsystem attempts to cold start the operating system, the diagnostic supervisor, or BOOT58 after a power failure or macro code halt instruction in three cases:

1. When the POWER ON ACTION switch is set to BOOT and the cold start flag is enabled.

2. When the user issues a boot command on the console terminal.

3. When the POWER ON ACTION switch is set to RESTART/BOOT, the warm start flag is disabled, and the cold start flag is cleared.

4.3.1 Cold Start Sequence

For automatic, unattended bootstrapping, the boot medium (disk pack or TU58 tape cartridge) must be located on drive 0 attached to either UB10 or MBAO, or on the console TU58. Bootstrapping under control of the boot command or the BOOT58 program allows the user to specify any system device and any channel adapter. Figure 4-1 shows the sequence of events and the various options available on a VAX-11/750 system boot.

4.3.1.1 Console Subsystem Action On Boot – At the beginning of the boot process, the console subsystem examines the BOOT DEVICE switch. Each position selects one of the four 256-byte ROMs that contain device specific boot block loading code. The console subsystem then performs the following functions.

- Initialize the cold start flag.

- Micro Verify.

- Initialize the PSL.

- Locate at least 64K contiguous bytes of page aligned usable memory to be used during bootstrapping.

- Initialize UBI0 by setting the UBI INIT bit.

- Load and validate at least the first 128 UBI0 map registers to address the first 64K bytes of usable memory.

- Transfer boot ROM into memory (base + ∧XFA00).

- Check the cold start flag. If it is set, halt. If it is not set, set it. This prevents cold start looping.

- Load input arguments to be used by the device ROM code and by VMB.

- Case on the boot ROM selected by the BOOT DEVICE switch.

- Check for nonexistent ROM; halt if there is no ROM in the selected socket.

The general registers receive the input arguments from the console subsystem.

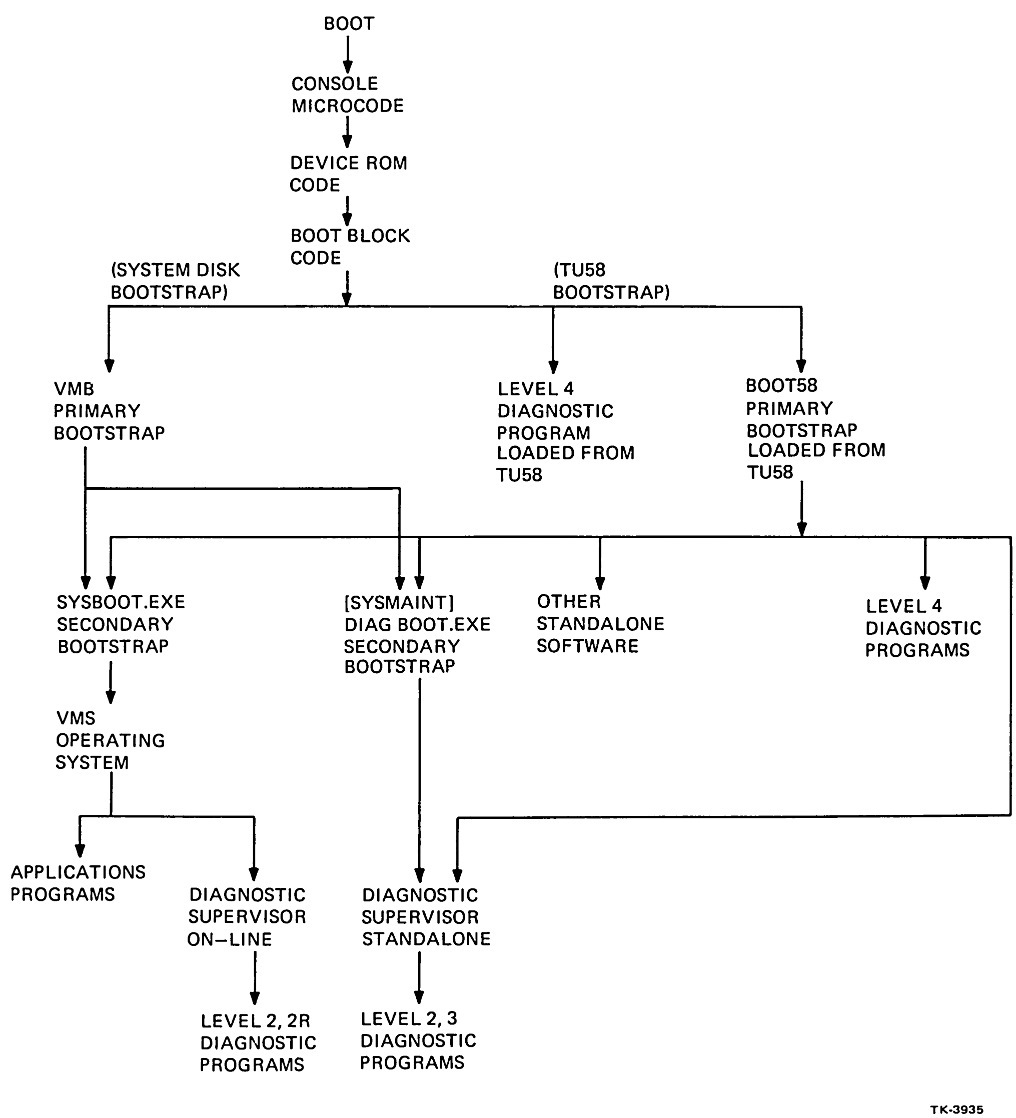


Figure 4-1 Boot Control Flow and Options

4.3.1.2 Device ROM Code Function on Boot – The console subsystem then transfers control to address 2 in the selected ROM code in main memory to begin macro instruction execution of the ROM code. Table 4-1 shows the four starting addresses.

Table 4-1 Device ROM Code Starting Addresses

Device ROM	Starting Address
A	FA02
B	FB02
C	FC02
D	FD02

Each device ROM code consists of four items.

1. The first two bytes, which identify the type of device the ROM supports.
2. A main routine starting at address 2.
3. A device driver routine.
4. The last byte of each ROM, a one-byte checksum that allows the console subsystem and memory diagnostic program to verify the ROM's contents.

The main routine gains control from the console subsystem.

The main routine then calls the driver routine to read the 512-byte boot block (boot 0) from the device into physical memory starting at the base of good memory.

For UNIBUS devices, the ROM driver routine assumes that the UBI map registers have been initialized by the console microcode to point to the first good 64K bytes of memory. Map Register 0 addresses the base physical address. Map Register 1 addresses the base physical address plus ∧X200, and so on. The driver routine also assumes that the map registers are marked as valid, and that each indicates a buffered data path.

For MASSBUS devices, the driver routine makes no such assumptions. It sets up an MBA map register before beginning to load the boot block into memory.

The driver routine then returns control to the main routine, leaving the status (success or failure) in R0. The symbol SS$_NORMAL indicates success. Zero, or a value with bit 0 cleared, indicates failure. If the driver routine has been successful, the main routine continues by setting up general registers as inputs to the boot block code and the primary bootstrap. It then transfers control to address C within the boot block code.

If the driver routine has failed to transfer the boot block successfully, the main routine executes a halt instruction. The console subsystem then prints out the PC and the halt code (6). If the keyswitch is in the LOCAL or REMOTE position, the console enters the console I/O mode and prints a prompt symbol on the terminal.

4.3.1.3 Boot Block Code Operation – The boot block contains device independent data and code. It depends on the driver routine in the boot ROM for code and parameters which are unique to each device.

Longword 0 contains the size of the primary bootstrap in blocks.

Longword 1 contains the swapped logical block number of the first block of the primary bootstrap.

Longword 2 contains the memory address to receive the primary bootstrap. The boot block code loads this file into memory starting at the base physical address plus 200(X). The boot block code calls the driver routine in the ROM code once for each block to accomplish the transfer. Then the boot block code transfers control to byte 0 of the loaded file, 200(X)(SP).

In the case of a TU58 boot, the boot process is complete at this point. Neither VMB nor a secondary bootstrap file is needed.

4.3.1.4 VMB Operation – In the normal disk bootstrapping sequence, the boot block code loads VMB and transfers control to VMB, the primary bootstrap for VMS and for the diagnostic supervisor (standalone). VMB performs the following functions.

1. Create a temporary System Control Block (SCB) to be used during bootstrapping.

2. Create and store restart data in a VMS system data structure called the Restart Parameter Block (RPB).

3. Identify all bus adapters and memory in the configuration.

4. Test all memory; mark every good page in a memory bit map.

5. Initialize the boot device's adapter.

6. If the boot command specified the solicit flag (bit 0 in R5), prompt for a secondary file specification.

7. Locate the secondary boot file; load the file into physical memory.

8. Transfer control to the secondary boot file.

4.3.1.5 SYSBOOT and VMS Operation – SYSBOOT is the secondary bootstrap program for VMS. It loads VMS into memory and transfers control to the VMS transfer address.

The VMS initialization code then starts up. One of the last steps in the VMS initialization process is the issuing of two commands to the console subsystem through the TXDB register. These commands enable both the warm and cold start flags. The new flag settings permit later unattended recovery from a power failure.

4.3.1.6 DIAGBOOT and Supervisor Operation – If you set the software boot control flag number 4 on the boot command to boot the diagnostic supervisor, VMB loads and starts DIAGBOOT instead of SYSBOOT. DIAGBOOT is the secondary bootstrap for the diagnostic supervisor. It determines the CPU type, loads the supervisor into memory, and transfers control to the initialization code in the supervisor.

The supervisor does not handle power failure or crash recovery. It therefore leaves the warm and cold start flags untouched.

4.4 BOOT58 OPERATION

BOOT58 is a supplementary bootstrap command processor. Like VMB, it is a primary bootstrap program. It enables you to perform the following functions.

- Load and start level 4 diagnostic programs.
- Bootstrap from the Massbus adapter.
- Bootstrap via downline loads.
- Bootstrap from a disk whose boot block is bad.
- Bootstrap from a disk whose error rate prohibits ROM and boot block loading of a primary bootstrap.
- Boot the diagnostic supervisor instead of VMS.
- Deposit and examine data in physical memory, general registers, and internal processor registers.
- Load and start a program from a magtape drive on a Massbus.
- Store and invoke indirect command files on the TU58 cartridge to perform any of the above functions automatically as well as interactively.

When the console subsystem transfers control to the device ROM code for the TU58, the ROM code loads logical block number (LBN) 0 from the TU58 cartridge inserted in the drive. This boot block contains the size and location of the file BOOT58.EXE (instead of the size and location of VMB.EXE). The boot block code in turn loads BOOT58, starting at the base of memory plus C000(X) and extending for approximately 4K bytes. When BOOT58 gains control, it prompts for input with the BOOT58> symbol.

4.4.1 Booting the BOOT58 Command Processor

There are four ways to bring up BOOT58. First, insert the BOOT58 tape cartridge in the TU58 slot.

1. Set the POWER ON ACTION switch to the BOOT position. Set the BOOT DEVICE switch on position D to select the boot ROM for the TU58. Turn on the machine, setting the keyswitch to the LOCAL position. Or, with the switch in LOCAL, press the RESET button.

2. Set the POWER ON ACTION to the HALT position. Turn on the machine, setting the keyswitch to the LOCAL position. Or, with the switch in LOCAL, press the RESET button. Type the boot command on the console terminal as follows.

   B DDA0<CR>

   DDA0 specifies the TU58 tape drive.

3. Set the POWER ON ACTION switch to the HALT position. Set the BOOT DEVICE switch to select the boot ROM for the TU58. Turn on power by setting the keyswitch to the LOCAL position. Or, with the switch in LOCAL, press the RESET button. Then type the boot command on the console terminal as follows.

   B<CR>

4. If the cache is not working and the situation is critical, you can boot BOOT58 after disabling the cache according to the procedure shown in Examples 5-2 and 5-3 in Chapter 5. Once BOOT58 is running, you can use it to load and start any program, leaving the cache disabled.

A BOOT58 command consists of a string of characters ending in a carriage return. If the last character before the carriage return is a hyphen (–), BOOT58 treats the hyphen as a line continuation character, and prompts for a continuation of the input command on the next line.

BOOT58 treats the exclamation point (!), as a comment delimiter. The program ignores all characters typed following the exclamation.

You may not deposit data or load programs into the memory space used by BOOT58 [from base address plus 200(X) to base address plus C000(X)]. BOOT58 responds to such an attempt with an error message.

4.4.2 BOOT58 Commands

BOOT [<devspec>]
: Boot from the device specified. If no device is specified, boot from the default boot device. This command cannot be used within an indirect command file.

DEPOSIT/<qualifier> <location> <value>
: Deposit <value> at the location specified by <location>. The <location> is interpreted according to <qualifier>.

EXAMINE/<qualifier> <location>
: Display the contents of <location>, where <location> is interpreted according to <qualifier>.

HELP
: Print this text at the console. This command cannot be used within an indirect command file.

LOAD <filespec>[/START:<value>]
: Load a file from the boot device into memory starting at the address specified <value>. If no starting location is specified, load the file beginning at the first free location in memory.

START <value>
: Initialize CPU and Jump to <value>.

@ <filespec>
: Execute command procedure from tape.

NOTE

You can abbreviate all TU58 commands to the first character.

4.4.3 BOOT58 Command Parameters

<location>	<value> ! <register>
<value>	<number> ! <shorthand>
<number>	Any nonnegative hexadecimal number.
<register>	0..F ! PSL
<shorthand>	Any one of the following: * = use last <location> specified + = use (last <location>) +1 – = use (last <location>)–1 0 = use contents of (last <location>)
<qualifier>	Any one of the following: P = physical memory address G = general register I = internal processor register

W = size = word
L = size = longword

<devspec> A device spec of the form: ddcu, where
dd = generic device type (e.g., DB)
c = controller designator (A ! B)
u = unit number (0..9)

<filespec> A legal RT-11 filename of the form: name.typ, where
name = any alphanumeric string of not more than 6 characters.
typ = any alphanumeric string of not more than 3 characters.
A null typ is acceptable.

```
BOOT58>E/L/P FF                     ! Examine the longword at physical
                                    ! location FF.
     P  000000FF  FFFFFFFF
BOOT58>D/W/P * 1                    ! Deposit 1 (word data type)in the
                                    ! previously referenced location.
BOOT58>E/L/P FF                     ! Examine the longword at the
                                    ! previously referenced location.
     P  000000FF  FFFF0001
BOOT58>
```

Example 4-2 Boot58 Examine and Deposit Commands

4.5 BOOTING FROM THE TU58 WITHOUT BOOT58

You can boot certain programs from the TU58 tape drive without bringing up BOOT58 first

EVKAA CPU Hard-core Instruction test
ECKAL VAX-11/750 Cache/Translation Buffer Test
ECSAA Diagnostic Supervisor

1. Insert the TU58 cartridge containing the required program file into the TU58 tape drive.

2. Bring up the console subsystem in the console I/O mode.

3. When the console subsystem responds with the console prompt, >>>, type
B DDA0<CR>

4. The console subsystem will load and start the program.

Note that you cannot boot all programs stored on TU58 tape cartridges in this manner. The boot block on the tape cartridge must specify the location and size of the program to be booted. Therefore, only one program per tape cartridge can be booted directly with the console boot command. However, if you boot BOOT58 or the diagnostic supervisor first, you can load any number of programs from a TU58 tape cartridge.

4.6 WARM START

The VMS operating system is capable of restarting itself following recovery from a power failure (if battery backup is installed). VMS will not restart (warm start) itself after a software crash. It may

invoke a boot from the device specified by the BOOT DEVICE switch, however. Standalone programs such as the diagnostic supervisor and BOOT58 do not restart themselves.

4.6.1 Console Subsystem Action On A Warm Start

When the console subsystem gains control of the VAX-11/750 system following a power failure, it examines the BOOT SELECT switch. The console subsystem attempts to restart VMS in either of two cases.

1. When the BOOT SELECT switch is set to the RESTART/HALT position and the warm start flag is enabled.

2. When the BOOT SELECT switch is set to the RESTART/BOOT position and the warm start flag is enabled.

If the BOOT SELECT switch is set to either RESTART/HALT or RESTART/BOOT, the console subsystem searches through physical memory for the restart parameter block (RPB). This block contains enough data for VMS to restart itself from the point at which the power failure occurred. When VMS crashes, it invalidates the RPB on the way down.

The first longword of the RPB contains its own address, and it is page-aligned. The console subsystem therefore searches through memory for a page-aligned longword that contains its own address. The console subsystem may reference the first four longwords of the RPB, shown in Figure 4-2.

PHYSICAL ADDRESS OF THE RPB		0:
PHYSICAL ADDRESS OF THE VMS RESTART ROUTINE		4:
CHECKSUM OF ^XIF LONGWORDS OF RESTART ROUTINE		8:
	WARM START FLAG (BIT 0)	C:

TK-4306

Figure 4-2 Restart Parameter Block, First Four Longwords

NOTE
warm start flag = 0 :enabled
warm start flag = 1 :disabled

If the console subsystem finds a longword which qualifies as the beginning of the RPB, it compares the longword at RPB + 0 with the longword at RPB + 4. If these longwords are equal, the RPB is not valid. If these longwords are unequal, the console subsystem compares the checksum of the first 1F(X) longwords of the restart routine against the contents of the third longword in the RPB. If the checksum is valid, the RPB is valid. The console subsystem then checks the warm start flag in the fourth longword.

If the console subsystem does not find a longword that points to itself on a page boundary, or the checksum is not valid, the console subsystem goes to the next page in memory to find the RPB. However, if the warm start flag is disabled, the warm start attempt has failed. The console subsystem then either boots the system or halts, according to the position of the POWER ON ACTION switch. If the RPB is found with a valid checksum, and the warm start flag is enabled, the console subsystem performs three further functions.

- Load the stack pointer (SP) with the address of the RPB plus 200(X).
- Load the argument pointer (AP) with the code for the microcode detected HALT.
- Start execution of the VMS restart routine, whose address is located at the second longword of the RPB.

4.6.2 VMS Action On A Warm Start

The VMS restart routine attempts to recover from the power failure by recreating a consistent software environment. The restart routine disables the warm start flag. Disabling this flag prevents warm start looping by showing that a warm start is in progress. The restart routine then examines the code in the AP that shows the nature of the microcode detected HALT.

If the code in the AP indicates a power failure and recovery, the restart routine continues the attempt to restart VMS. It initiates power recovery operations such as reinitialization of I/O controllers, cancellation or retry of active I/O operations, and so on. If the recovery succeeds, VMS executes an instruction requesting the console subsystem to reenable the warm start flag.

If the code in the AP indicates that the system crashed, or, if for any reason, the restart procedure cannot recover, VMS checks the bugreboot flag. Bugreboot is a SYSGEN parameter. If the flag is set, VMS executes a macrocode HALT instruction. The console subsystem then gains control. It either attempts a cold start or keeps the CPU halted, according to the POWER ON ACTION switch. If the console subsystem keeps the CPU halted and the keyswitch is in LOCAL or REMOTE, the console enters the console I/O mode and prompts for input.

If VMS checks the bugreboot flag and finds it cleared, and the executive debugger is enabled, the VMS code stops at a breakpoint. If the bugreboot flag is cleared and the executive debugger is disabled, VMS hangs in a tight loop.

BLANK

CHAPTER 5
RUNNING LEVEL 4 DIAGNOSTIC PROGRAMS

Two level 4 diagnostic programs apply to the VAX-11/750:
EVKAA, Hard-core Instruction Test
ECKAL, Cache/Translation Buffer Test

5.1 BOOTING LEVEL 4 DIAGNOSTIC PROGRAMS

DIGITAL ships each of these programs with a boot block on its own TU58 tape cartridge. To load and start a program from one of these dedicated tapes, proceed as follows.

1. Insert the cartridge in the TU58 slot in the front panel.
2. Turn the POWER ON ACTION switch to the HALT position.
3. Type CTRL/P on the console terminal.
4. Type B DDAO<CR> on the console terminal.

The program should start and run continuously.

```
^P                          ! CTRL/P.
>>>B  DDAØ                  ! Boot the program
                            ! (e.g., EVKAA)
                            ! from the TU58.
%%                          ! Micro Verify success.
EVKAA-5.Ø  done!            ! First pass done.
EVKAA-5.Ø  done!            ! Second pass done.
     .
     .
     .
```

Example 5-1 Booting a Level 4 Diagnostic Program

Before the system loads the program, the console microcode scans memory to find a good 64 kilobyte section of memory beginning on a page boundary. If the console microcode finds a good 64 kilobyte section, it loads the program into that section. Programs that are position independent (PIC) execute at any starting address. However, programs which are not PIC fail unless they are loaded beginning at address 0. The cache/TB test is not PIC. It checks to see if it has been relocated and halts displaying the error PC if it has been. The system hangs before or after printing error messages.

If there is a problem in the cache which prevents program execution, you can boot any program, that does not enable the cache automatically, from the TU58, with the cache disabled, in either of two ways. If one of the four boot ROM sockets is empty, follow the procedure shown in Example 5-2. If all four sockets contain boot ROMs, follow the procedure shown in Example 5-3.

Programs you can boot with the cache disabled.
- Level 4 diagnostic programs
- Diagnostic supervisor
- BOOT58

```
! Set the BOOT DEVICE switch to
! point to an empty boot ROM socket.

>>>I                                    ! Initialize the CPU
>>>B                                    ! Boot.
%%                                      ! Micro Verify success.

0000FD00  14                            ! CPU Halt; nonexistent
                                        ! ROM.
>>>D/I  25  1                           ! Disable the cache.
>>>D/G  F  F20402                       ! Set the PC to point to
                                        ! the TU58 boot ROM.
>>>C                                    ! Start the boot process.

EVKAA-5.0  done!                        ! The hard-core instruction
EVKAA-5.0  done!                        ! test is running continuously.
   .
   .
   .
```

Example 5-2 Booting a Level 4 Program with the Cache Disabled, Using an Empty Boot ROM Socket

```
>>>I                                    ! Initialize the CPU.
>>>D/I  25  1                           ! Disable the cache.
>>>D/G  0  0                            ! Initialize general registers.
>>>D/G  1  F28000
>>>D/G  2  FFE000
>>>D/G  3  0
>>>D/G  E  200
>>>D/G  F  F20402                       ! Set the PC to point to
                                        ! the TU58 boot ROM.
>>>C                                    ! Start the boot process.

EVKAA-5.0  done!                        ! The hard-core instruction
EVKAA-5.0  done!                        ! test is running continuously.
   .
   .
   .
```

Example 5-3 Booting a Level 4 Program with the Cache Disabled, When There Is No Empty Boot ROM Socket

5.2 LEVEL 4 PROGRAM ERROR INTERPRETATION AND LOOP CONTROL

When a level 4 program detects a hardware failure, it executes a HALT instruction. In response, the console microcode prints the current PC and the halt code on the console terminal. If the halt code is

06, it indicates that the processor executed a HALT instruction. Other halt codes indicate that the program is not running properly. See Table 2-5 in Chapter 2 for a list of the halt codes and their meanings.

If the halt code is 06, you can look in the listing for the instruction corresponding to the failing PC. Note that if the base address for the program is not 0, you must find the base address and subtract it from the PC in order to find the corresponding instruction in the listing. Examine register 10 in order to find the base address, as follows.

```
>>>E/G 10      ! Examine register R10.
```

Consider the case where an operator runs the hard-core instruction test and gets the following console output.

```
>>>B DDA0                        ! Boot the program from the
                                 ! TU58.
%%                               ! Micro Verify success.
0000CD90  06                     ! Updated PC and halt code.
>>>
```

Example 5-4 Halt in the Hard-Core Instruction Test

Look up location CD8F (the PC minus 1) in the program listing. Read the test description and analyze the code. If the fault is a hard error, you can cause the program to loop by replacing the HALT instruction with a NOP instruction. Examine the location first to be sure you are at the right place. Then replace 00 (HALT) with 01 (NOP).

```
>>>E/B   CD8F                    ! Examine location CD8F.
    P        0000CD8F    00
>>>D   *   01                    ! Deposit 01 in the byte at
                                 ! location CD8F.
>>>C                             ! Continue. The program
                                 ! should loop on the error.
```

Example 5-5 Looping on a Hard Error in a Level 4 Diagnostic Program

However, if the error is intermittent, the program will progress out of the loop on the first success. In order to loop every time, you must find the instruction which checks for success or failure (See Example 5-6).

```
CD36  13203   T18_S1:  MOVW    #1, SUBNUM
CD3D  13204            CLRL    R8
CD3F  13205            CLRL    R5
CD41  13206   10$:     MOVL    INSVSRC[R8],R3
CD49  13207            MOVL    INSVPOS[R8],R4
CD51  13208            MOV8    INSVSIZ[R8],R5
CD59  13209            CLRQ    R6
CD5B  13210            BICPSW  #15
CD5D  13211            BISPSW  R8
CD5F  13212            INSV    R3,R4,R5,R6
CD62
CD64  13213            MOVPSL  R1
CD66  13214            BICL    #^XFFFFFFFE,R1
CD6D  13215            BLBC    R8,20$
CD70  13216            MOVL    #1,R0
CD73  13217            BRB     30$
CD75  13218   20$:     CLRL    R0
CD77  13219   30$:     XORL3   R0,R1,R2          ; Compare expected,
                                                 ; received data
CD7A
CD7B  13220            BEQL    40$               ; Test for success.
CD7D  13221            MOVAB   1000$,ERROR_PC
CD88  13222            MOVB    #1,ERROR_FLAG
CD8F  13223   1000$:   HALT                      ; Halt.
CD90  13224            BRB     10$               ; LOOP. PC Printed.
CO92  13225   40$:                               ; Next part of test.
```

Example 5-6 Level 4 Program Listing Sample EVKAA, Test 18, Subtest 1

In this case, the BEQL 40$ instruction at line 13220 checks for success of the operation under test. Two bytes correspond to this instruction. Replace them both with 01, as shown in Example 5-7.

```
>>>E/W   CD7B            ! Examine the two
                         ! bytes at CD7B.

   P     0000CD7B            1513
>>>D  *  0101            ! Deposit 2 NOP codes.
>>>E/B  CD8F             ! Examine the HALT location.

   P     0000CD8F            00
>>>D  *  01              ! Replace the HALT
                         ! with a NOP.
>>>C                     ! Continue. The
                         ! program should loop
                         ! on the error.
```

Example 5-7 Setting up a Loop on an Intermittent Error

CHAPTER 6
PRIMARY AND SECONDARY LOAD PATHS

System disks provide the primary load paths for diagnostic programs. All VAX-11/750 computer systems use UNIBUS based disks or MASSBUS based disks.

To run a level 2 or level 3 diagnostic program in the standalone mode (without VMS) you should load the diagnostic supervisor, and the program to run under it, from the system disk. Any level 2 or level 3 diagnostic program will run under the supervisor in the standalone mode.

However, if your VAX-11/750 system has a hardware failure in the disk subsystem or the CPU cluster, you may be unable to load programs from a system disk. The TU58 tape drive provides a secondary load path.

To run diagnostics in the on-line mode (with VMS), boot the operating system first, and then run the diagnostic supervisor.

6.1 PRIMARY LOAD PATH BOOT (STANDALONE)

Boot the diagnostic supervisor and load and start diagnostic programs from a system disk as follows.

- Press the RESET button on the front panel.
- Type a boot command, in response to the console prompt:
  B[/X]/10<space><ddcu><CR>
  <ddcu> specifies the disk from which to boot the supervisor. /10 specifies the diagnostic boot control flag to VMB.

  If [SYSMAINT] is not the directory you require, type
  >>>B/200 <ddcu>.
  VMB will then prompt you with
  BOOTFILE:
  type
  [<directory-name>]<file-name>

  NOTE

  You must include square brackets around the directory name.

  For example:
  [TEST] ECSAA.EXE

- When the supervisor starts and prompts with DS>, invoke the configuration file to define the hardware configuration to the supervisor by typing @CONFIG.

- Use the run command to load and start a diagnostic program.

See Chapters 4 and 5 of the *VAX Diagnostic System User's Guide* for details on supervisor commands. See Table 2-4 in Chapter 2 for the boot device codes.

```
>>>B/10   DLA0            ! Boot the supervisor from the RL02
                          ! drive 0 on controller A.


DIAGNOSTIC SUPERVISOR.  ZZ-ECSAA  5.02-99  1 JAN 1980 00:00:00.00

DS> @CONFIG               ! Define the system
                          ! hardware configuration
                          ! to the supervisor.

DS> ATTACH  DW750 CMI DW0
DS> ATTACH RL11 DW0 DLA 774400 160 5
DS> ATTACH RL02 DLA DLA0
DS> ATTACH RK611 DW0 DMA 777440 210 5
DS> ATTACH RK07 DMA DMA0
DS> ATTACH RK07 DMA DMA1
DS> ATTACH RK07 DMA DMA2
DS> SHOW DEVICE
_DW0  DW750          40F30000
_DLA  RL11   _DW0    40FFF900  CSR=774400(0) VECTOR=000160(0) BR=5.
_DLA0 RL02   _DLA    00000000
_DLA1 RL02   _DLA    00000000
_DMA  RK611  _DW0    40FFFF20  CSR=777440(0) VECTOR=000210(0) BR=5.
_DMA0 RK07   _DMA    00000000
_DMA1 RK07   _DMA    00000000
_DMA2 RK07   _DMA    00000000
DS> @ <EOF>
DS> SELECT DW0                     ; Select DW0 for testing.
DS> RUN ECCBA                      ; Run the UBI
                                   ; diagnostic program.
```

Example 6-1 Booting the Supervisor Standalone from the Primary Load Path and Running a Diagnostic Program

6.2 SECONDARY LOAD PATH BOOT (STANDALONE)

If you cannot boot the diagnostic supervisor from a system disk, try to boot it from the TU58 tape drive, the secondary load path.

Use this secondary load path to load programs which test the hardware in the primary load path. These programs are called load path diagnostics.

The load path for VAX-11/750 systems includes the following hardware.

- Central processor
- Memory
- UNIBUS interface
- MASSBUS interface
- The disk drive containing the SYSMAINT directory or VMS
- A tape drive (if the system includes one)

DIGITAL supplies load path diagnostics on TU58 tape cartridges. Each level 4 program is on a tape cartridge containing only a boot block and the program EXE file.

Level 2 and level 3 programs are stored several to a tape cartridge. Each level 2 and level 3 program cartridge also contains a boot block and a copy of the supervisor. See EVNDX, the VAX Development MAINDEC Index, for a list of the load path diagnostic programs and tape cartridges that apply to each type of VAX-11/750 configuration.

Use the secondary load path as follows.

- Turn the POWER ON ACTION switch to the HALT position.
- Press the RESET button on the front panel.
- Insert the appropriate TU58 tape cartridge in the TU58 slot in the front panel.
- Type the boot command in response to the console prompt: B DDAO<CR>. DDAO specifies the TU58 tape drive.
- If the supervisor starts and prompts with DS>, use the attach command to define the hardware configuration for the supervisor. You must use the attach command several times in order to define the connection between the CPU and the device to be tested. See Paragraph 6.3 of this manual. The TU58 tape cartridges do not contain CONFIG files.
- If the supervisor does not start, run level 4 diagnostics according to the instructions in Chapter 5.
- Use the select command to select a device for testing. See the *VAX Diagnostic System User's Guide* for details.
- Then type in the run command to run a diagnostic program.

```
>>>B/10  DDA0                        ! Boot the supervisor
                                     ! from the TU58.

DIAGNOSTIC SUPERVISOR ZZ-ECSAA 5.02-99  1 JAN 1980 00:00:00.00

DS> ATTACH DW750 CMI DW0             ! Attach the UBI.
DS> ATTACH RL11 DW0 DLA 774400 160 5
                                     ! Attach the RL11 controller.
DS> ATTACH RL02 DLA DLA0             ! Attach the first RL02 disk
                                     ! drive.
DS> ATTACH RL02 DLA DLA1             ! Attach the second RL02 disk
                                     ! drive.
DS> SELECT DLA0, DLA1                ! Select the 2 RL02 disk
                                     ! drives.
DS> RUN EVRFA                        ! Run the RL02 repair
     .                               ! diagnostic program.
     .
     .
```

Example 6-2 Booting the Supervisor from the Secondary Load Path (TU58) and Running a Diagnostic Program

6.3 DIAGNOSTIC SUPERVISOR ATTACH COMMAND SPECIFIC TO THE VAX-11/750

ATTACH<uut-type><link-name><generic-device-name> . . .<CR>

When you load diagnostics from the SYSMAINT directory on a system disk, you can type @CONFIG to invoke the configuration command file. This file consists of a series of attach commands which define the hardware configuration for the supervisor, as shown in Example 6-1.

However, when you load diagnostics from the console TU58 tape drive and when you wish to test some devices not mentioned in the CONFIG file, you must use the attach command.

Use several attach commands, before starting a diagnostic program, to define each unit under test (UUT), and the devices that link it to the CMI, for the supervisor. If you are testing several units at once, repeat the attach command for each device. Every unit under test is uniquely defined by a hardware designation and a link.

The first parameter <uut-type> is the hardware designation of the unit under test. For example, RH750, TM03, TE16, and DZ11 are hardware designations.

The second parameter, <link-name>, is the name of the piece of hardware that links the unit under test, in most cases through intermediate links, to the CMI. For example, an RH750 is linked to the CMI; an MTa is linked to an RH750; a TU45 is linked to an MTa; and a DZ11 is linked to a DWn. You must attach each piece of hardware (with the exception of the CMI) before you can use it as a link in an attach command.

The third parameter is the generic device name, which identifies to the supervisor the particular unit to be tested. Use the form <GGan> for the device name. <GG> is a two-character generic device name (alphabetic). <a> is an alphabetic character, specifying the device controller. <n> is a decimal number in the range of 0–255, specifying the number of the unit with respect to the controller.

Use the unit number, <n> or <a>, only if it is applicable to the device. You must supply additional information for some types of hardware to enable the diagnostic program to address the device. For example, you must supply the controller number for a TM03, and the CSR vector and BR for a UNIBUS device. If you do not include additional information, but the information is necessary, the supervisor will prompt you for it (see Example 6-3).

Table 6-1 Device Naming Conventions

Type	Link	Generic	Additional Information
KA750	CMI	KAn	<G-floating><H-floating><WCS-last -address><time of year clock> <accelerator>
RH750	CMI	RHa	
DW750	CMI	DWa	
RP06	RHa	DBan	
RP05	RHa	DBan	
RP04	RHa	DBan	
RM03	RHa	DRan	
RK611	DWa	DMa	<ucsr><uvector><ubr>
RK07	DMa	DMan	
RK06	DMa	DMan	
TM03	RHa	MTa	<drive>
TE16	MTa	MTan	
TU45	MTa	MTan	
TU77	MTa	MTan	

Table 6-1 Device Naming Conventions (Cont)

Type	Link	Generic	Additional Information
DZ11	DWa	TTa	<ucsr> <uvector> <ubr> <EIA> <20MA>
DUP11	DWa	XJan	<ucsr> <uvector> <ubr>
DMC11	DWa	XMan	<ucsr> <uvector> <ubr>
KMC11	DWa	XMan	<ucsr> <uvector> <ubr>
LP11	DWa	LPa	<ucsr> <uvector> <ubr>
CR11	DWa	CRa	<ucsr> <uvector> <ubr>
DR11B	DWa	UZa	<ucsr> <uvector> <ubr>
PCL11	DWa	??a	<ucsr> <uvector> <ubr>
TS04	DWa	MSan	<ucsr> <uvector> <ubr>
RL02	DLa	DLan	
RL11	DWa	DLa	<ucsr> <uvector> <ubr>

The definitions for the additional fields are:

 	Adapter BR level	decimal	4–7
<drive>	Massbus drive	decimal	0–7
<ucsr>	Unibus CSR address	octal	760000–777776
<uvector>	Unibus vector	octal	2–776

In the generic name.

a is a letter from A to Z.
n is a decimal number in the range 0–255.
?? is a generic device name that may be any two letters.

```
DS> ATTACH DW750 CMI DW0        ! Attach the DW780.
DS> ATTACH DZ11 DW0 TTA         ! Attach the DZ11,TTA.
CSR? 760120                     ! The supervisor prompts
VECTOR? 320                     ! for information not
BR? 4                           ! supplied in the command
MODULE TYPE? EIA                ! line.
DS>
```

Example 6-3 Attach Command

6.4 RUNNING THE SUPERVISOR ON-LINE

You must boot the operating system before running the supervisor and diagnostic programs on-line (with VMS). If you have load path problems, run the supervisor in the standalone mode, booting it from the TU58, as explained in Paragraph 6.2. To run diagnostics on-line, proceed as follows.

- Boot VMS.
- Log into the field service account.
- In response to the VMS prompt, $, type RUN ECSAA.
- Invoke the configuration file to define the hardware configuration for the supervisor.
- Use the select command to select a device for testing.
- Use the run command to run the required diagnostic program.
  Only level 2 and 2R diagnostic programs run in the on-line mode.

```
      >>>B   DLA1                          ! Boot VMS from the
                                           ! RLØ2 drive 1 on
                                           ! controller A.
VAX/VMS Release 2.Ø                        1-JAN-198Ø ØØ:ØØ:ØØ:Ø.
      ! Log into the Field service account
      $ RUN ECSAA                          ! Run the diagnostic
                                           ! supervisor.
           .
           .
           .
DIAGNOSTIC SUPERVISOR.  ZZ-ECSAA-5.Ø2-99  1-JAN-198Ø  ØØ:ØØ:ØØ.ØØ

      DS>  @CONFIG                         ! Invoke  the  configuration
                                             file.
DS>   ATTACH  DW75Ø CMX DWØ
DS>   ATTACH RL11 DWØ DLA 7744ØØ 16Ø 5
DS>   ATTACH RLØ2 DLA DLAØ
DS>   ATTACH RK611 DWØ DMA 77744Ø 21Ø 5
DS>   ATTACH RKØ7 DMA DMAØ
DS>   ATTACH RKØ7 DMA DMA1
DS>   ATTACH RKØ7 DMA DMA2
DS>   SHOW DEVICE
_DWØ  DW75Ø        4ØF3ØØØØ
_DLA  RL11  _DWØ   4ØFFF9ØØ   CSR=7744ØØ(Ø) VECTOR=ØØØ16Ø(Ø) BR=5.
_DLAØ       RLØ2   _DLA       ØØØØØØØØ
_DLA1       RLØ2   _DLA       ØØØØØØØØ
_DMA  RK611 _DWØ   4ØFFFF2Ø   CSR=77744Ø(Ø) VECTOR=ØØØ21Ø(Ø) BR=5.
_DMAØ       RKØ7   _DMA       ØØØØØØØØ
_DMA1       RKØ7   _DMA       ØØØØØØØØ
_DMA2       RKØ7   _DMA       ØØØØØØØØ
DS> @ <EOF>
DS>  SELECT DLAØ                          ! Select an RLØ2 disk drive.

DS>  RUN EVRAA                            ! Run the disk reliability
                                           ! program.
```

Example 6-4 Running Diagnostics On-Line.

CHAPTER 7
BUILDING AND MAINTAINING THE DIAGNOSTIC SYSTEM DISK

7.1 BUILDING A DIAGNOSTIC DISK PACK

Each VAX-11/750 system requires a diagnostic disk pack as well as a VMS disk pack, or a combination VMS and diagnostic disk pack, for proper system operation.

DIGITAL distributes diagnostic system installation kits for VAX-11/750 systems in two package types. UNIBUS based VAX-11/750 systems are shipped with RK07 or RL02 disk packs, which already contain the diagnostic files. MASSBUS based VAX-11/750 systems are shipped with a magnetic tape containing the diagnostic files.

7.1.1 MASSBUS Single Disk Systems

For MASSBUS single disk based VAX-11/750 systems, you may build a diagnostic area [SYSMAINT] on the same disk pack as that which contains the VMS operating system.

Single disk systems are shipped with a full set of diagnostic files on TU58 cartridges. Transfer the files from the TU58 tape cartridges to the system disk pack according to the following procedures.

1. Ensure that VMS is running properly and type CTRL/Y to return control to the monitor.

2. Insert the DSC1 TU58 tape cartridge.

3. Type
   $ MCR FLX /RS/CO=CS1:ECUBA.COM/RT/FA
   to VMS.

4. Type
   $ @ECUBA
   to VMS.

5. Locate the first TU58 tape cartridge containing files to be transferred.

6. Insert the TU58 tape cartridge. In response to the prompt message,
   TYPE IN THE NAME OF THE MOUNTED TAPE CARTRIDGE [<CR>=EXIT],
   type in the name of the tape cartridge. After you have transferred the files from all the TU58 tape cartridges, type a carriage return following the prompt message to exit from the script. At this point, the SYSMAINT area on your system disk is complete.

7.1.2 MASSBUS Dual Disk Systems

For MASSBUS dual disk based systems, you must transfer the files on magnetic tape to a formatted disk pack on the system disk drive before you can run many of the macro level diagnostic programs. Build the diagnostic disk pack according to the following procedures.

1. Return control to the console I/O mode in the console program as follows.

   Type CTRL/P on the console terminal.

2. When the console program displays the prompt symbol, >>>, locate the DSC1 TU58 tape cartridge and insert it in the TU58 slot on the front panel. Place the BOOT DEVICE switch in the position corresponding to the TU58 boot ROM (position D).

3. Mount the diagnostic distribution kit magnetic tape on the MASSBUS tape drive 0 with the write-enable ring removed. Ensure that the tape is on-line and positioned at BOT.

4. Mount a formatted scratch disk pack in disk drive 0 and ready the drive for I/O. Type B. If the system is RPO6 based, follow Steps 5 and 6. If the system is RMO3 based, follow Step 7.

5. For an RPO6, obtain the list of bad blocks from the disk pack test data supplied by the manufacturer. The list identifies each bad block by cylinder, track, and sector coordinates. Convert this data to logical block numbers (LBN) using the following formula:

   LBN = (cylinder * 19 + track) * 22 + sector.

   You may run the READ ALL section of the disk formatter program (EVRAC) before starting the DSC process as an alternative to using the manufacturer's data. The disk formatter program provides the logical block numbers of the bad blocks on the disk pack.

6. In response to the DSC> prompt on the console terminal, type the following command to create and verify the RPO6 diagnostic disk.

```
DSC>DBAØ:/VE/BAD=MAN=MTAØ:/RW
```

Example 7-1 Creating an RPO6 Diagnostic Disk Pack

The DSC1 program then prompts you for bad block data with BAD=. Type in the logical block numbers of the bad blocks as shown below.

```
BAD=1376.<CR>
BAD=4591Ø.<CR>
BAD=<CR>                    ! Begin transfer.
```

Example 7-2 Entering Bad Block Numbers

NOTE
The dot (.) signifies a decimal number.

The DSC1 program begins transferring data after you respond with a carriage return to the BAD= prompt.

7. For an RMO3 based system, type the following command in response to the DSC> prompt.

```
DSC>DRAØ:/VE= MTAØ:/RW
```

Example 7-3 Creating an RMO3 Diagnostic Disk Pack

DSC1 copies the files on the magnetic tape to the RMO3 disk pack.

8. At the end of the transfer to either the RMO3 or the RPO6 disk pack, the DSC1 program rewinds the tape for a verification pass. The /VE qualifier in the command line specifies this pass.

   Example 7-4 Shows the console terminal output.

```
>>>B

VAX/VMS DSC-1, VERSION 1.Ø  9-MAY-198Ø

DSC>DMAØ:/VE=MTAØ:/RW

DSC -- 45 STARTING VERIFY PASS

DSC>                          ! Type <CR> to exit.

>>>
```

Example 7-4 Transferring Diagnostic Files from Magnetic Tape to Disk with DSC1

9. Create a boot block on the newly created disk pack under VMS with the WRITEBOOT utility at first possible moment.

10. Boot the diagnostic supervisor as explained in Paragraph 4.4 of Chapter 4.

11. You may be unable to build a diagnostic pack with the DSC tape or unable to boot the supervisor because of a hardware failure. In either case, use the set of load path TU58 tape cartridges to load the diagnostic supervisor and run the appropriate diagnostic programs. See Chapter 6 for details.

12. On dual-drive systems, you may copy the [SYSMAINT] files to the VMS system pack. To perform this transfer, place the VMS disk pack on drive 0, place the diagnsotic disk pack on drive 1, and boot VMS. Then use the VMS copy command to transfer the diagnostic files to the VMS pack.

7.2 UPDATING DIAGNOSTIC FILES

When you receive a set of TU58 tape cartridges to update the diagnostic files, transfer the new files from the tape cartridges to the diagnostic system disk pack according to the following procedures.

1. Ensure that VMS is running properly and type a CTRL/Y on a terminal to return control to the monitor.

2. Locate tape cartridge containing the ECUBB file and insert it in the slot.

3. Type
   $ MCR FLX/RS/CO=CS1:ECUBB.COM/RT/FA
   to VMS.

4. Type
   $ @ECUBB
   to VMS.

5. Locate the tape cartridge containing the files to be transferred. Insert the tape cartridge. In response to the prompt message,
   TYPE THE NAME OF MOUNTED TAPE CARTRIDGE [<CR>=EXIT],
   type in the name of the tape cartridge. The ECUBB script deletes files to be replaced, transfers all of the diagnostic files on the TU58 tape cartridge, and then prompts the operator for another tape cartridge. After you have transferred the files from all the update tape cartridges, type a carriage return following the prompt message to exit from the ECUBB script. At this point your diagnostic system disk is complete.

CHAPTER 8
VAX-11/750 TROUBLESHOOTING GUIDELINES

The following guidelines are intended to help the field service engineer isolate operational faults in a VAX-11/750 system by using the diagnostic tools available. You may find these guidelines helpful if the problem in the system you are troubleshooting falls into one of the three following categories:

1. The system is covered by a remote diagnosis maintenance contract and the DIGITAL Diagnostic Center (DDC) has conducted a diagnosis session which has pointed out a faulty subsystem. (Go to Section 1.)

2. The VAX/VMS operating system runs, but a customer application does not run or system performance is degraded. (Go to Section 2.)

3. The VAX/VMS operating system will not run, or you suspect a faulty hardware component but are not sure which specific subsystem is at fault. (Go to Section 3.)

These guidelines are intended only to get you started on a logical fault isolation and repair process. Knowledge and experience on the VAX-11/750 are assumed. Remember that technical support personnel at the DDC are always available to provide telephone assistance.

1. The DDC has isolated the problem to a faulty subsystem.

   a. Rerun the diagnostic which gave the fault indication for the DDC.

   b. Run any further diagnostics necessary to isolate the fault to the failing gate array or module. If the faulty module is one which has gate arrays, microdiagnostics run under RDM control will isolate the fault to a gate array string. Refer to Section 6 for hints on running microdiagnostics.

   c. Replace the indicated gate array(s) or module and run diagnostics again to verify the fix.

   d. If gate array replacement did not fix the problem, replace the module and run diagnostics to verify the fix.

   e. If module replacement did not fix the problem, you should contact support personnel.

NOTE

For L0001, L0002, L0003, L0004, L0007, and L0011 modules, the primary maintenance philosophy calls for fault isolation to the gate array and repair by gate array replacement. If after exercising this primary philosophy the problem is still present, the engineer is expected to fix the CPU by module replacement.

f. Once you have fixed the problem and verified the fix with micro and/or macrodiagnostics, boot up the VAX/VMS operating system and perform system operation verification with the UETP.

g. When you are satisfied that the system is ready to be returned to the customer, notify the DDC of your repair action.

2. VMS runs, but application does not or system performance is degraded.

a. Use the SYE utility to generate an error log report. Analyzing this report may help you isolate the problem to a hardware subsystem.

b. If the SYE report points to a subsystem, then run the appropriate diagnostics to further isolate the fault and take the necessary corrective measures.

c. If the SYE report does not seem to point to any particular hardware subsystem, the problem may be in software and you should contact support personnel for assistance. Depending on the nature of the customer's problem, the support personnel may have to consult with corporate VMS software groups.

3. VMS does not run or some unspecific hardware fault is suspected.

a. Try to boot the diagnostic supervisor from the system disk.

b. If the supervisor fails to load and run, the fault is either in the "primary load path" (system disk and associated controller and channel) or in the CPU. Go to Section 4.

c. After successful supervisor boot, begin subsystem fault isolation by the process of elimination, starting with the CPU as follows.

d. Run diagnostics against main memory (ECKAM), CPU architecture (EVKAB, EVKAC, EVKAD, EVKAE), and VAX-11/750 CPU specifics (ECKAX).

NOTE

ECKAM assumes that the first memory array is good. To fully check the first array, swap it with a memory module in another slot and rerun the ECKAM diagnostic.

If these run error free, go to item e, otherwise, interpret the error reports and take further diagnostic (microdiagnostics) action and/or corrective action as necessary. Refer to Section 1, items b through f above.

e. Run diagnostics against the major subsystems in your system:

- UNIBUS and UNIBUS mass storage (ECCBA and device/controller diagnostics)
- MASSBUS and MASSBUS mass storage (ECCAA and device specific diagnostics)

Interpret error reports and take appropriate corrective action. If this diagnostic approach yields no meaningful error indications, you should consider contacting support personnel for assistance.

f. Once you have fixed the problem and verified the fix with micro and/or macrodiagnostics, boot up the VAX/VMS operating system and perform system operation verification with the UETP.

4. Cannot boot supervisor through primary load path.

   a. Interpret microverify printout from boot attempt. (Refer to the *VAX-11/750 Diagnostic Mini Reference Guide* (EK-KC750-RM).)

      If the microverify printout is normal, go to item b. If microverify gave an error code, a CPU fault is indicated; use microdiagnostics to further isolate the fault. Refer to Section 1, items b through f.

      If console printout is nonexistent or unintelligible, the console terminal or console interface may be at fault. Take diagnostic measures to verify terminal operation. The console interface hardware resides on the L0004 module. However, for the CPU console to operate, the CPU data paths, sequencing logic, and control store must have some level of integrity. If the CPU data paths, sequencing logic, or control store is defective, the operation of the RDM console will not be affected. By typing <CTRL>P <CTRL>D, with the RDM properly installed, the RDM console is invoked and you can proceed with your diagnosis using microdiagnostics.

   b. Try to boot supervisor through the secondary load path (TU58). If the boot is successful, go to item c. Otherwise, a CPU problem is indicated; go to Section 5.

   c. Run diagnostics against main memory (ECKAM), CPU architecture (EVKAB, EVKAC, EVKAD, EVKAE), and VAX-11/750 CPU specifics (ECKAX). If these run error free, go to item d, otherwise interpret the error reports and take further diagnostic (microdiagnostics) action and/or corrective action as necessary. Refer to Section 1, items b through f.

   d. Run the channel diagnostic against the channel which is the primary load path for your system (UNIBUS run ECCBA, MASSBUS run ECCAA). If the channel diagnostic runs error free, go to item e, otherwise interpret the error reports and take the appropriate corrective action.

   e. Run device and controller diagnostics against the load path disk subsystem. Interpret the error reports and take the appropriate corrective action. If at this point your diagnostic actions have not yielded meaningful information, you should contact support personnel for assistance.

   f. Once you have fixed the problem and verified the fix with micro and/or macrodiagnostics, boot up the VAX/VMS operating system and perform system operation verification with the UETP.

5. Cannot boot diagnostic supervisor through the secondary load path.

   a. Perform a CCS parity scan using the RDM. (Refer to Section 7.) If the parity scan passes, go to item b. If the parity scan failed, the control store is probably faulty, so replace the L0005 module and verify the fix.

b. Boot the hardcore diagnostic (EVKAA) from TU58. If the diagnostic loads and runs without an error halt, go to item c, otherwise consult the EVKAA listing for error halt meaning and take further diagnostic measures (microdiagnostics) to isolate the faulty gate array or module. Refer to Section 1, items b through f above.

NOTE
If during the EVKAA run the CPU gets a machine check or other exception, it will halt in a trap catcher block.

c. Boot the cache translation buffer diagnostic (ECKAL) from TU58. If this diagnostic loads and runs error free, you should contact support personnel for assistance. A failure of this diagnostic usually indicates a fault on the L0003 module.

NOTE
Part of ECKAL (test 13) assumes that the UBI and UET are somewhat functional.

Consult the ECKAL listing for error halt meaning and take further diagnostic measures (microdiagnostics) to isolate the faulty gate array. Refer to Section 1, items b through f above.

d. Once you have fixed the problem and verified the fix with micro and/or macrodiagnostics, boot up the VAX/VMS operating system and perform system operation verification with the UETP.

6. Hints on running VAX-11/750 microdiagnostics.

   a. In order to run micros on the VAX-11/750, the L0006 (RDM) module must be installed in slot 6 of the CPU and the console and TU58 signal connectors on the back of slot 6 must be moved to the right-hand pin column (as viewed from the rear of the cabinet) of slot 6. If the machine under test is covered by a remote diagnosis maintenance contract, this will already have been done.

   b. Initialize the CPU to the HALT state. You should get the microverify codes, a halt code, and the CPU console prompt (>>>).

   c. Invoke the RDM console monitor by typing <CONTROL>D. You should get the RDM prompt (RDM>).

   d. Insert the microdiagnostic TU58 tape cassette into the TU58 transport and type:

      TE/C <RET>

      The tape should load and you should get the micromonitor (MICMON) prompt (MIC>).

   e. Set the MICMON TRACE flag by typing:

      SE FL TR <RET>

      This will cause the name of each micro test to be printed as the microdiagnostic runs.

f. Run the microdiagnostic by typing:

   DI <RET>

   The tape should begin moving and test names will be printed. When an error is encountered, an error report will be printed telling the nature of the failure and suspected faulty gate arrays and modules in order of probability.

g. The microdiagnostic will stop and control will return to MIC after reporting the first error. To run the remainder of the failing test and subsequent tests, thus giving you more error reports, you may clear the HALT flag by typing:

   CL FL HA <RET>

   and continue by typing:

   CO <RET>

7. Performing an RDM parity scan.

   a. Prepare for RDM console operation. (Refer to Section 6, items a through c.)

   b. Stop the CPU clocks by typing:

      STO <RET>

   c. To start the parity scan from CCS address 0000, type:

      PAR 0000 <RET>

   d. A parity error at CCS address 17FD is normal. If this is the only CCS parity printout, the parity scan passed. The RDM parity scan function will only report the first parity error it encounters. To scan beyond the first error, simply repeat items b and c, but instead of specifying address 0000, specify an address higher than the first error address. The parity scan can only detect hard CCS parity errors; a soft error will typically cause a machine check exception.

The above sequences on microdiagnostic running and parity scan are only intended to get you started. For information on RDM and MICMON commands and RDM error codes, refer to the pocket size *VAX-11/750 Mini Reference Guide* (EK-KC750-RM). For more in-depth information on RDM and MICMON, refer to the *Remote Diagnosis Technical Manual* (EK-KC750-TM).

BLANK

VAX-11/750 DIAGNOSTIC SYSTEM
OVERVIEW MANUAL

Reader's Comments

EK-VXD75-UG-002

Your comments and suggestions will help us in our continuous effort to improve the quality and usefulness of our publications.

What is your general reaction to this manual? In your judgment is it complete, accurate, well organized, well written, etc.? Is it easy to use? ______

What features are most useful? ______

What faults or errors have you found in the manual? ______

Does this manual satisfy the need you think it was intended to satisfy? ______

Does it satisfy *your* needs? ______ Why? ______

☐ Please send me the current copy of the *Technical Documentation Catalog*, which contains information on the remainder of DIGITAL's technical documentation.

Name ______ Street ______
Title ______ City ______
Company ______ State/Country ______
Department ______ Zip ______

Additional copies of this document are available from:

Digital Equipment Corporation
444 Whitney Street
Northboro, MA 01532
Attention: Printing and Circulating Service (NR2/M15)
Customer Services Section

Order No. EK-VXD75-UG

Fold Here

Do Not Tear – Fold Here and Staple

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD, MA.

POSTAGE WILL BE PAID BY ADDRESSEE

Digital Equipment Corporation
Communications Development and Publishing
1925 Andover Street
Tewksbury, Massachusetts 01876

BLANK